Departamento de Ingeniería Telemática

PROYECTO FIN DE CARRERA

Estudio del consumo de energía en un dispositivo Android

Autor: Elena López Orgaz

Tutor: Pablo Serrano Yañez-Mingot

Director: Carlos Jesús Bernardos Cano

Leganés, octubre de 2012

Título: Estudio del consumo de energía en un dispositivo Android
Autor: Elena López Orgaz
Tutor: Pablo Serrano Yañez-Mingot
Director: Carlos Jesús Bernardos Cano

EL TRIBUNAL

Presidente: Jaime J. García Reinoso

Vocal: Matilde P. Sánchez Fernández

Secretario: Alicia Rodríguez Carrión

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 8 de Octubre de 2012 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Gracias a todas las personas que me han apoyado durante estos años y han hecho posible que haya llegado hasta aquí.

Resumen

Las nuevas funcionalidades y la integración de diferentes tecnologías de comunicación en los nuevos dispositivos móviles hacen que el consumo de potencia se dispare, en unos dispositivos que disponen de una fuente de energía muy limitada. Como consecuencia, ha aumentado la preocupación sobre cómo administrar y ahorrar energía.

Entender y caracterizar el consumo de energía en un dispositivo móvil es necesario para poder desarrollar métodos de administración de energía que permitan reducir el consumo. Con este objetivo, el presente proyecto fin de carrera lleva a cabo un estudio sobre el consumo de energía en un dispositivo con sistema operativo Android. Android, creado por Google y de código abierto, es uno de los sistemas operativos para dispositivos móviles con mayor penetración en el mercado. Además de ofrecer gran funcionalidad y un potente entorno de desarrollo, Android implementa un mecanismo de administración de energía con el objetivo de reducir el consumo cuando el dispositivo no está siendo utilizado.

Se ha desarrollado una metodología que permite obtener el consumo de distintos componentes hardware del dispositivo bajo estudio. Los resultados experimentales obtenidos permiten definir qué componentes son los que descargan a mayor velocidad la batería y consumen niveles de potencia más altos. Ya que la mayor parte de la potencia consumida se atribuye a las tecnologías de comunicación, se lleva a cabo un análisis más detallado sobre el consumo de las tecnologías Wi-Fi y 3G.

Palabras clave: smartphone, consumo energía, Android, Wi-Fi, 3G

Abstract

The new functionalities and the integration of different communication technologies increase drastically the power consumption in new mobile devices, which are powered with a limited source of energy. As a result, the awareness about how to manage and save energy has increased.

Understanding and profiling the energy consumption in a mobile device is necessary to develop new methods for energy management in order to reduce power consumption. To this end, this final degree project carries out a study about the energy consumption in a device with Android operating system. Android, developed by Google, is one of the operating systems for mobile devices with a higher market penetration. As well as providing a large functionality and a powerful development environment, Android implements an energy management mechanism which has the goal of reducing power consumption when the device is not being used.

We have developed a methodology to obtain the power consumption by different device hardware components. The experimental results show which components drain the battery faster and consume high power levels. Since the main part of the power consumption is attributed to communication technologies, we have carried out a more detailed analysis about energy consumption of Wi-Fi and 3G technologies.

Keywords: smartphone, energy consumption, Android, Wi-Fi, 3G

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1 Motivación del proyecto

La evolución de los teléfonos móviles en los últimos años se ha producido a pasos agigantados. En la última década, hemos pasado de utilizar terminales que solo envían y reciben llamadas y SMS, a dispositivos que navegan por Internet, reproducen audio y realizan fotos y videos. Estos nuevos teléfonos móviles se denominan teléfonos inteligentes o *smartphones*.

Un smartphone es un teléfono móvil que integra la funcionalidad de los teléfonos móviles convencionales junto con capacidades hasta ahora propias de los PCs. La inteligencia de estos dispositivos, aparte de su avanzado hardware, reside en los sistemas operativos diseñados para dispositivos móviles. Entre estos nuevos sistemas operativos encontramos Symbian, Windows Mobile, iOS o Android. Desde el punto de vista hardware, los smartphones llevan integrados una gran variedad de componentes como CPU, memoria, cámara, acelerómetro, pantalla LCD, interfaz radio, Wi-Fi, Bluetooth, GPS, micrófono, altavoces, etc.

Para ofrecer las nuevas funcionalidades, en estos nuevos teléfonos móviles han adquirido vital importancia las tecnologías de comunicación como Bluetooth, Wi-Fi, 3G o GPS. Para las comunicaciones de datos, las tecnologías más utilizadas son 3G y

Wi-Fi. Según un estudio sobre el uso de smartphones, elaborado por la firma Analys Mason [1] en mayo 2012, el 75% de los usuarios usaron Wi-Fi en sus smartphones, un 82% usaron 3G, y un 64% utilizaron ambas redes. Pero por otro lado, el uso de las tecnologías de comunicación hace que la potencia consumida se dispare en unos dispositivos con una fuente de energía de duración bastante limitada.

La velocidad a la que ha evolucionado la capacidad de las baterías no ha sido la misma a la que han evolucionado las funcionalidades de los teléfonos móviles. La gran mayoría de smartphones utilizan baterías recargables electroquímicas, normalmente de iones de litio (Li-ion). El problema de estas baterías es su corta duración cuando se mantienen conexiones a redes de datos. Este inconveniente ha planteado que los propios fabricantes de dispositivos tengan que diseñar tanto el hardware como el software con la premisa de reducir los niveles de potencia consumidos por estos terminales. Como consecuencia de no poder aumentar la capacidad de las baterías, es necesario que los propios dispositivos reduzcan los niveles de potencia consumida.

La nueva conciencia de reducir el consumo de energía de los dispositivos móviles ha generado la necesidad de nuevas formas de administración de energía. Para los ordenadores de sobremesa la administración de energía se usaba para no disparar la potencia consumida y evitar sobrecalentamientos. Sin embargo, la energía en los dispositivos móviles se ha convertido en un recurso clave y los sistemas operativos tienen que tratar con esta limitación, para reducir el consumo y alargar la vida de las baterías.

Android es un ejemplo de sistema operativo que ha desarrollado una nueva administración de energía. Aunque se basa en Linux, que es un sistema operativo no destinado a terminales móviles, Android ha implementado, con la idea de que la energía es un recurso limitado, una política de administración de energía mucho más agresiva. Este mecanismo optimiza sobre todo la energía consumida por los dispositivos cuando no están siendo utilizados.

Realizar una eficiente administración de energía requiere tener un buen conocimiento previo sobre dónde y cómo la energía se consume. Analizar qué partes del dispositivo hacen que las baterías se descarguen rápidamente puede ayudar a desarrollar esquemas de eficiencia energética. Sin embargo, obtener el consumo de cada uno de los componentes de un smartphone no es algo trivial. Sobre esta problemática se ha desarrollado el presente proyecto: desarrollar una metodología que permita obtener y evaluar el consumo de diferentes componentes en un smartphone.

1.2 Introducción al estudio del consumo de energía en dispositivos móviles

Con la proliferación de los dispositivos móviles, se han llevado a cabo diferentes estudios que centran su problemática en analizar el consumo de energía de estos dispositivos y proponen modelos matemáticos para estimar este consumo [2], [3], [4], [5].

La inquietud por conocer cuánto consume un terminal móvil aparece ya con los ordenadores portátiles, interesándose la mayoría de trabajos por el consumo asociado a las comunicaciones de datos. En uno de estos trabajos [2], se analiza el consumo de energía de la interfaz IEEE 802.111 en distintos modos de operación y se proporciona evidencia sobre como la tasa del tráfico transmitido afecta al patrón de descarga de la batería.

En cuanto a los smartphones, existen estudios previos que evalúan el consumo de energía de distintos componentes, en los que se demuestra que gran parte de la energía consumida en los dispositivos móviles se debe a las comunicaciones. Un ejemplo es el estudio [6], en el que se atribuye la mayor parte de la potencia consumida al modulo GSM y al *display* (pantalla táctil LCD, acelerador de gráficos e iluminación). En otro estudio [3], llevado a cabo sobre un smartphone Nokia con la herramienta Nokia Energy Profiler, se obtiene que 3G consume significativamente más energía respecto a GSM y Wi-Fi. En el trabajo [5], se plantea un modelo con el que se obtienen niveles de potencia mayores para Wi-Fi que para 3G. También se obtienen resultados sobre algunos componentes que, como la memoria RAM o SD, tienen un efecto mínimo en la potencia total consumida [6]. Por otro lado, se evalúan algunas estrategias de administración de energía, como apagar la iluminación de la pantalla durante una llamada que emplea Android, que puede ahorrar hasta un 40% de potencia [6].

La mayoría de estos trabajos presentan algún tipo de problemática. Ciertos modelos diseñados para estimar la potencia consumida por diferentes componentes de un smartphone solo son válidos para dispositivos del mismo tipo a los que se han utilizado para el estudio [5]. Por otro lado, en estos trabajos, las medidas son realizadas con equipos electrónicos específicos capaces de obtener la potencia consumida por el dispositivo en tiempo real. Sin embargo, este tipo de medidas son difíciles de reproducir para medir el nivel de potencia consumida por diferentes componentes de forma aislada.

Aparte de estudios como los mencionados, se han desarrollado herramientas para medir el consumo en distintos dispositivos móviles. Para Symbian, existe la ya mencionada aplicación Nokia Energy Profiler [7], que permite monitorizar el consumo de energía de las aplicaciones en tiempo real. Entre las posibles opciones, se puede

[1] Aunque IEEE 802.11 y Wi-Fi no son términos equivalentes, nos referiremos a ellos indistintamente. En el capítulo 2, se explica el significado de ambos términos y la relación entre ambos.

obtener la potencia en un tiempo determinado de medida, tal y como se puede ver en la Figura 1. La limitación de esta herramienta es que mide la potencia consumida por todo el teléfono cuando se utiliza una determinada aplicación, no siendo posible conocer que parte de esa potencia se debe por ejemplo a las conexiones de las redes datos. Además, la propia aplicación de medida tiene un consumo considerable de potencia.

Figura 1. Potencia medida con la aplicación Nokia Energy Profiler [7]

En Android existe una herramienta integrada en el sistema operativo que monitoriza el uso de la batería. Muestra gráficos (ver Figura 2) que detallan en forma porcentaje la contribución de componentes y aplicaciones, a la energía total consumida desde el último ciclo de carga. Por lo que esta herramienta es limitada a la hora de obtener el consumo del dispositivo en tiempo real. Es útil para identificar aplicaciones con un uso exhaustivo de la batería.

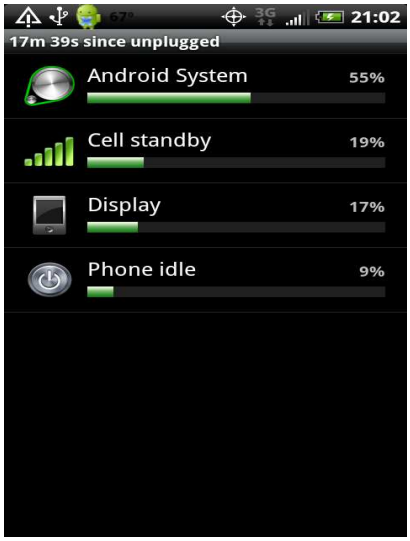

Figura 2. Herramienta de monitorización del uso de la batería incluida en Android

En el mercado de aplicaciones de Android, se puede encontrar la aplicación PowerTutor [8], que muestra la potencia consumida por los principales componentes principales como CPU, interfaces de comunicación, pantalla, GPS y por las aplicaciones utilizadas. Esta aplicación realmente no mide la potencia consumida por cada componente, sino que la estima en base al modelo desarrollado en el estudio [5] ya mencionado, pero que solo es válido para los dispositivos utilizados para derivar el modelo, de forma que para otros dispositivos los resultados pueden ser diferentes.

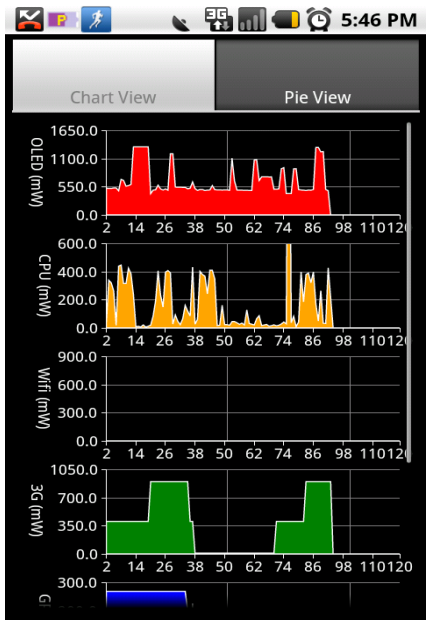

Figura 3. Una de las vistas de la aplicación PowerTutor

En base a la problemática y a las limitaciones planteadas por los estudios previos que se han analizado y las herramientas que encontramos para obtener el consumo de energía en dispositivos móviles, se plantea para este proyecto los objetivos que se describen a continuación.

1.3 Objetivos

El objetivo fundamental de este proyecto es el estudio de la energía consumida por los componentes principales de un dispositivo móvil Android. Es de especial interés, evaluar el consumo de las dos tecnologías de comunicación más utilizadas en terminales móviles actualmente: Wi-Fi y 3G.

En un primer lugar, se pretende desarrollar una metodología con la que se pueda medir no solo la potencia consumida por el dispositivo en su conjunto, sino principalmente obtener el consumo de diferentes componentes por separado.

Una vez obtenidos los valores de potencia consumida por distintos componentes, el objetivo es realizar un análisis de los mismos y poder determinar cuáles consumen más energía. En cuanto a las tecnologías de comunicación, además se va a estudiar en más detalle el consumo en sus distintos modos de operación. Estos resultados pueden ayudar, tanto a los usuarios para hacer un uso eficiente de un smartphone, como a desarrolladores a la hora de realizar el diseño del software y el hardware de estos dispositivos.

El último objetivo de este proyecto es determinar la viabilidad de la metodología desarrollada para caracterizar el consumo de un dispositivo móvil mediante las funcionalidades del sistema operativo Android y sin haber empleado un hardware de medida especializado.

1.4 Estructura de la memoria

En este primer capítulo se ha descrito la motivación y los objetivos del proyecto, junto a una introducción al tema que se va a tratar a lo largo de esta memoria. A continuación se incluye como se estructura el resto de la memoria junto con un breve resumen de cada capítulo, para facilitar la lectura de este documento.

La memoria se estructura en los siguientes capítulos:

Capítulo 2. Estado del arte: Se describe la plataforma Android sobre la que se ha trabajado, las tecnologías de comunicación que se evalúan, y los conceptos generales de consumo de energía en dispositivos móviles.

Capítulo 3. Metodología: Se plantea la metodología empleada para evaluar la potencia consumida en un smartphone Android, exponiendo los objetivos a cumplir y justificando cómo se ha desarrollado.

Capítulo 4. Medidas de consumo: En este capítulo se presenta las medidas obtenidas mediante la metodología descrita en el capítulo 3 y se validan realizando una comparación con los resultados obtenidos en otros estudios.

Capítulo 5. Conclusiones y trabajo futuro: Por último, se exponen las conclusiones a las que se ha llegado tras la realización de este proyecto y se explica el trabajo futuro que se puede realizar para completar los resultados obtenidos.

Capítulo 6. Planificación y presupuesto: Se detalla las fases del desarrollo del proyecto y se incluye una descripción de los diferentes costes para obtener el presupuesto total de proyecto.

Referencias: Se detalla la bibliografía empleada para desarrollar algunos puntos del proyecto, pudiéndose consultar por parte del lector para ampliar la información.

Anexos: Al final de esta memoria se incluye una serie de anexos donde se entra más en detalle sobre cómo se han realizado algunas tareas del proyecto.

- *Anexo A: Instalación y configuración del SDK y el entorno de desarrollo*: Es una guía sobre como configurar e instalar los paquetes necesarios para trabajar con Android y el entorno de desarrollo.

- *Anexo B: Herramienta ADB (Android Debug Bridge):* Se explica las configuraciones necesarias para utilizar la herramienta ADB y las operaciones que se permiten realizar con ella.

- *Anexo C: Instalación de la herramienta SL4A:* Es una guía para la instalación de la herramienta SL4A y los componentes necesarios para utilizarla.

- *Anexo D: Acceso root HTC Legend:* Se describen los requisitos y los pasos necesarios para tener acceso como root al smartphone HTC Legend.

- *Anexo E: Configuración del escenario de trabajo Wi-Fi:* Se describen los pasos que se han seguido para la configuración del escenario de trabajo con la tecnología de comunicación Wi-Fi.

- *Anexo F: Código aplicación BatteryMonitor*: Se incluye el código completo de la aplicación que realiza la monitorización del estado de la batería.

Capítulo 2

Estado del arte

Este capítulo presenta las tecnologías, las herramientas y los conceptos que han sido necesarios para el desarrollo de este proyecto.

El primer punto está dedicado a la plataforma Android, que es el sistema operativo sobre el que se ha realizado el estudio del consumo de energía de un dispositivo móvil. Por un lado, se presenta las características y la arquitectura de Android y por otro lado, se explica cómo se realiza la administración de energía en los dispositivos con este sistema operativo. Además, se presenta cómo funciona el scripting en Android.

En segundo lugar, se da una visión sobre el consumo de energía en los terminales móviles, describiendo las fuentes de energía (baterías) utilizadas por este tipo dispositivos y cómo es en general el consumo de energía en un dispositivo móvil y las formas de evaluarlo

El último punto presenta dos tecnologías de comunicación en dispositivos móviles: Wi-Fi y 3G. Estas dos tecnologías son las que se han elegido para estudiar consumo de energía en un smartphone Android. Se mostrará que ambas tecnologías son muy diferentes, al igual que la administración de energía de cada una de ellas.

2.1 Plataforma Android

Android [9] es una plataforma software de código abierto diseñada para dispositivos móviles y desarrollada por Google y la Open Handset Alliance.[2] Incluye un sistema operativo basado en Linux, un *middleware* y una capa de aplicaciones. Además, el *SDK* (*Software Development Kit*) de Android proporciona las herramientas y las *API* (*Application Programming Interface*) necesarias para desarrollar aplicaciones Android empleando el lenguaje de programación Java.

En 2011, Android fue el sistema operativo para dispositivos móviles con mayor cuota de mercado (ver Figura 4), superando a iOS, el sistema operativo de Apple para terminales móviles. Gran parte de su éxito se debe a que es sistema operativo libre, gratuito, multiplataforma, que intenta optimizar al máximo el uso de los recursos de los dispositivos.

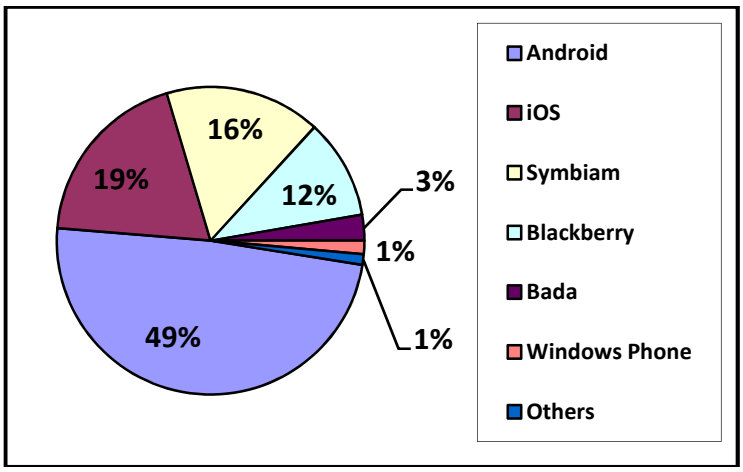

Figura 4. Cuota mundial de mercado de smartphones en 2011 según Canalys [10]

2.1.1 Características técnicas de Android

A continuación se detallan las características principales de la plataforma Android [9]:

- Sistema operativo basado en Linux.
- Incluye un **SDK (Software Development Kit)** con las herramientas necesarias para el desarrollo de aplicaciones utilizando el lenguaje de programación Java.

[2]La Open Handset Alliance es una agrupación comercial formada por 84 compañías de tecnología que se dedica a desarrollar estándares abiertos para dispositivos móviles. http://www.openhandsetalliance.com/

- **Máquina virtual Dalvik:** Es la máquina virtual donde se ejecutan las aplicaciones. Está optimizada para dispositivos móviles de manera que requiere poca memoria y se puedan usar varias instancias simultáneamente sin que el dispositivo se ralentice.

- **Navegador integrado Webkit**: Es un navegador de código abierto *WebKit* y actúa como base para varias aplicaciones que hay actualmente en el mercado.

- **Marco de aplicación** que permite la reutilización y el remplazo de componentes, facilitando el desarrollo de aplicaciones.

- **Gráficos optimizados** gracias a una librería 2D mejorada. Los gráficos 3D están basados en la especificación OpenGL ES 1.0 (aceleración hardware opcional).

- **SQLite:** Android incluye en el SDK el sistema SQLite y lo utiliza para el almacenamiento de datos estructurados.

- **Soporte multimedia** para formatos comunes de audio, vídeo e imagen (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF).

- Dependiendo del hardware:

  - **Telefonía GSM**

  - **Bluetooth, EDGE, 3G y Wi-Fi**

  - **Cámara, GPS, brújula, y acelerómetro**

- **Potente entorno de desarrollo** que incluye un emulador de dispositivos, herramientas para la depuración, perfiles de memoria y rendimiento, y un *plu-gin* para el entorno integrado de desarrollo de Eclipse (IDE).

2.1.2 Arquitectura

Android presenta una arquitectura en la que sus componentes se agrupan en capas, como se puede observar en la Figura 25, de modo que cada capa utiliza los elementos de la capa inferior para realizar sus funciones. Este tipo de estructura se denomina pila.

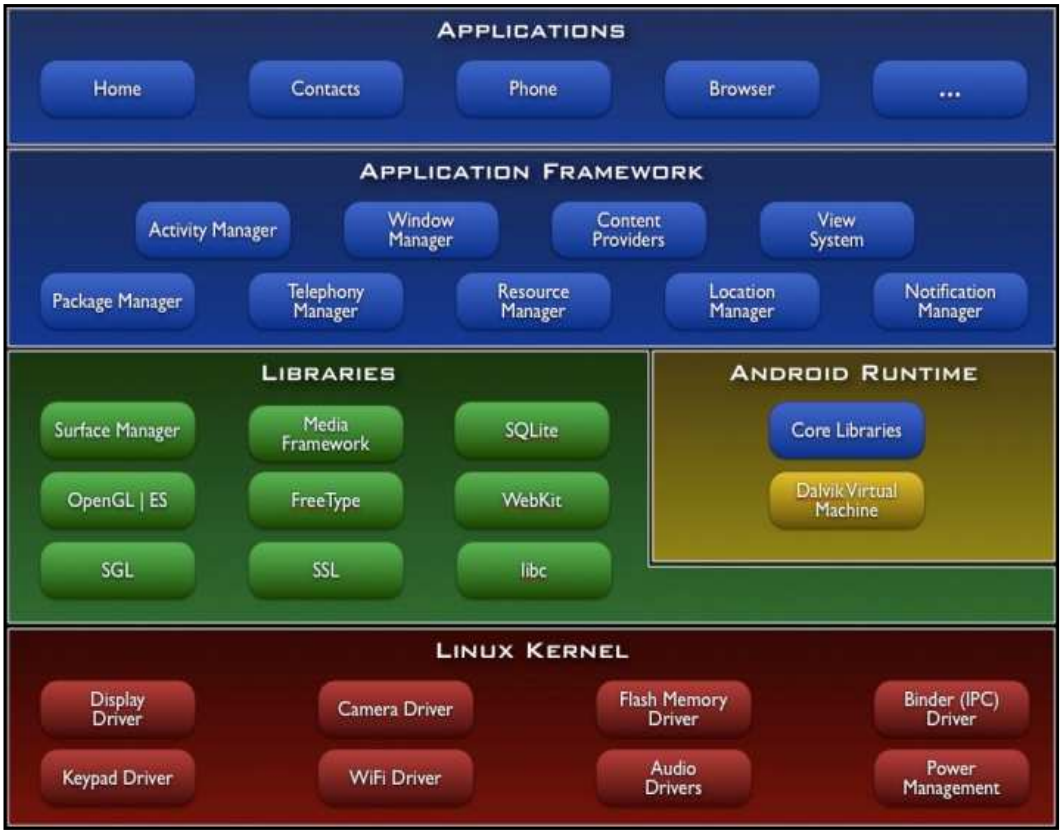

Figura 5. Arquitectura del sistema operativo Android [9]

De abajo hacia arriba de la pila, los componentes que forman la arquitectura de Android son los siguientes:

- ***Linux Kernel***

El núcleo del sistema operativo está basado en un *kernel* de Linux versión 2.6, similar cualquier distribución Linux, solo que adaptado a las características hardware de los dispositivos móviles. Este kernel gestiona los servicios principales como la seguridad, los elementos de comunicación, los controladores hardware y la administración de energía, de memoria y de procesos. Además actúa como un nivel de abstracción entre el hardware y el resto de capas software de la arquitectura.

- ***Librerías***

Sobre el kernel se sitúa una capa con las librerías nativas de Android. Están desarrolladas en C/C+++ y compiladas por la arquitectura hardware de cada dispositivo. Normalmente es el fabricante quien se encarga de desarrollarlas e instalarlas en el terminal.

El objetivo principal de las librerías es proporcionar a las aplicaciones funcionalidades para las tareas que se repiten frecuentemente, sin que se tengan que codificar cada vez que se necesiten. Sus capacidades se ofrecen a los desarrolladores a través del marco de aplicación de Android

Algunas de las librerías más importantes son: OpenGL (motor gráfico), bibliotecas multimedia (formatos de audio, imagen y video), Webkit (navegador), SSL (cifrado de comunicaciones), FreeType (fuentes de texto), SQLite (base de datos), entre otras.

- ***Entorno de ejecución***

Tal y como se puede apreciar en la Figura 5, el entorno de ejecución de Android no se considera como una capa en sí misma ya que está formada por un conjunto de librerías. Estas librerías ofrecen las funcionalidades principales de las librerías del lenguaje de programación Java.

El componente principal del entorno de ejecución es la maquina virtual Dalvik. Cada aplicación en Android se ejecuta en su propio proceso, con una instancia propia de la máquina virtual Dalvik. Dalvik está diseñado de tal forma que un dispositivo puede ejecutar varias máquinas virtuales al mismo tiempo de forma eficiente, sin interferir unas con otras.

Las aplicaciones se escriben en lenguaje Java y son compiladas de forma que se genera un ejecutable binario compatible con la arquitectura hardware específica de los dispositivos Android. Los ejecutables se generan con el SDK de Android, y no se pueden ejecutar en máquinas virtuales Java convencionales. Durante el proceso de compilación, se genera de forma intermedia, el *bytecode* habitual (archivo .class) y en el proceso final se convierten al formato específico de Davilk (archivo .dex). Los archivos ejecutables de Dalvik están optimizados para ocupar el mínimo espacio de memoria.

La máquina virtual Dalvik tiene acceso a las librerías mencionadas anteriormente y mediante ellas accede a las funcionalidades ofrecidas por el kernel de Linux.

- ***Marco de aplicación***

En esta capa se incluyen todas las clases y los servicios que utilizan directamente las aplicaciones para realizar sus funciones. La mayoría de los componentes de esta capa son librerías Java que acceden a los recursos mediante la máquina virtual Dalvik. Las más importantes, que aparecen en el diagrama de la arquitectura (Figura 5), se detallan a continuación:

- *Administrador de actividades (Activity Manager)*: Se encarga de controlar el ciclo de vida de las actividades y de la propia vida de las actividades. Las actividades son las ventanas que muestran las aplicaciones en las pantallas de los dispositivos Android, al igual que las ventanas de un PC.

- *Administrador de ventanas (Windows Manager)*: Se encarga de organizar lo que se muestra en pantalla, creando las superficies que pasarán a ser ocupadas por las actividades.

- *Proveedor de contenidos (Content Provider)*: Permite encapsular un conjunto de datos para ser compartidos entre las aplicaciones para tener control de como se accede a la información.

- *Vistas (Views)*: Las vistas son los controles que se incluyen dentro de las interfaces de usuario de las aplicaciones, como botones, cuadros de texto, listas, etc… y otras más sofisticadas como un navegador web o un visor de Google Maps.

- *Administrador de paquetes (Package Manager)*: Las aplicaciones Android se distribuyen en paquetes (archivos .apk), que contiene los archivos .dex junto con los recursos y archivos adicionales que necesite la aplicación. Esta librería permite obtener información sobre los paquetes instalados y gestionar la instalación de nuevos paquetes.

- *Administrador de telefonía (Telephony Manager)*: Proporcionas acceso a la pila hardware de telefonía del dispositivo, permitiendo realizar llamadas o enviar y recibir SMS/MMS.

- *Administrador de recursos (Resource Manager)*: Permite gestionar todos los elementos propios de una aplicación que se incluyen directamente en el código: cadenas de texto, imágenes, sonidos y disposiciones de las vistas dentro de una actividad (layout).

- *Administrador de ubicaciones (Location Manager)*: Permite determinar la posición geográfica del dispositivo mediante GPS o redes disponibles y trabajar con mapas.

- *Administrador de notificaciones (Notification Manager*): Proporciona los servicios para notificar al usuario cuando algo requiere su atención, mostrando alertas en la barra de estado, emitiendo sonidos o activando el vibrador.

- ***Aplicaciones***

En la capa más alta de la arquitectura se encuentran las aplicaciones. Entre ellas se incluyen tanto las aplicaciones incluidas de serie en el dispositivo como las instaladas por el usuario, las que tienen interfaz de usuario y las que no, y las nativas (programadas en C o C++).

2.1.3 Administración de energía en Android

La administración de energía en los sistemas operativos destinados a dispositivos móviles es cada vez más necesaria debido al gran consumo de recursos de estos equipos y a la capacidad limitada de las baterías.

El sistema operativo Android realiza una administración de energía, sobre el estándar de administración de energía de Linux, con una política mucho más agresiva de gestión y ahorro de energía. Se basa en la premisa de que la CPU no consume si no hay aplicaciones ni servicios que necesiten energía [11].

El mecanismo de administración de energía en Android, está basado en *wakelocks*, para dinámicamente controlar el consumo de energía. Un wakelock es una función del servicio PowerManager (ver Figura 7) que permite controlar el estado de energía del dispositivo. Las aplicaciones y componentes tienen que crear y adquirir wakelocks para mantener los recursos activos. Si no hay ningún wakelock activo, la CPU se apaga y se pasa a un estado de bajo consumo. Esta funcionalidad si se usa de forma correcta, permite un significativo ahorro de energía en los dispositivos cuando no están siendo utilizados.

En la Figura 6, se muestra la máquina de estados que representa el modelo de administración de energía descrito. Hay tres estados “SLEEP”, “NOTIFICATION”, y “AWAKE”. Cuando una aplicación obtiene un wakelock completo o se produce un evento por actividad de la pantalla o el teclado, el dispositivo se mantiene o se mueve al estado “AWAKE”. Si hay un *timeout* o se presiona el botón de apagado/encendido, se produce la transición al estado “NOTIFICATION”. Mientras se adquiera un wakelock parcial el dispositivo se mantiene en el estado “NOTIFICATION”. Cuando todos los wakelocks parciales se liberan, se pasa al estado “SLEEP”. En este estado, si todos los recursos se activan, ocurre la transición al estado “AWAKE”.

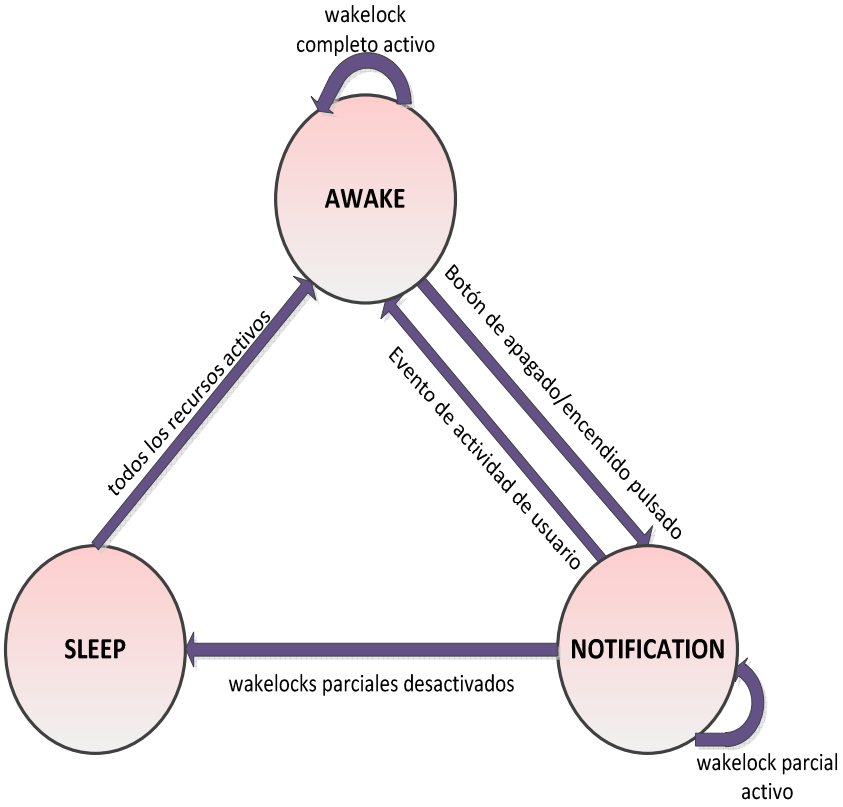

Figura 6. Máquina de estados representando la administración de energía Android

En conclusión, si no hay ningún wakelock activo, Android suspende la CPU y pasa al estado “SLEEP”, que es un estado de bajo consumo. Es necesario aclarar que cuando la pantalla se apaga, no supone que se pase a este modo ya que la CPU puede seguir trabajando. Con la pantalla apagada, por ejemplo se puede estar escuchando música o descargando archivos. Por lo tanto, este estado de bajo consumo se produce cuando la CPU no está activa. El consumo es pequeño ya que el nivel de actividad es muy bajo, asociado a las comunicaciones para mantener al dispositivo conectado a la red para recibir llamadas y mensajes. Ya que este estado es dominante en el tiempo en el que dispositivo está encendido, es crucial la potencia consumida en este estado para la vida de la batería.

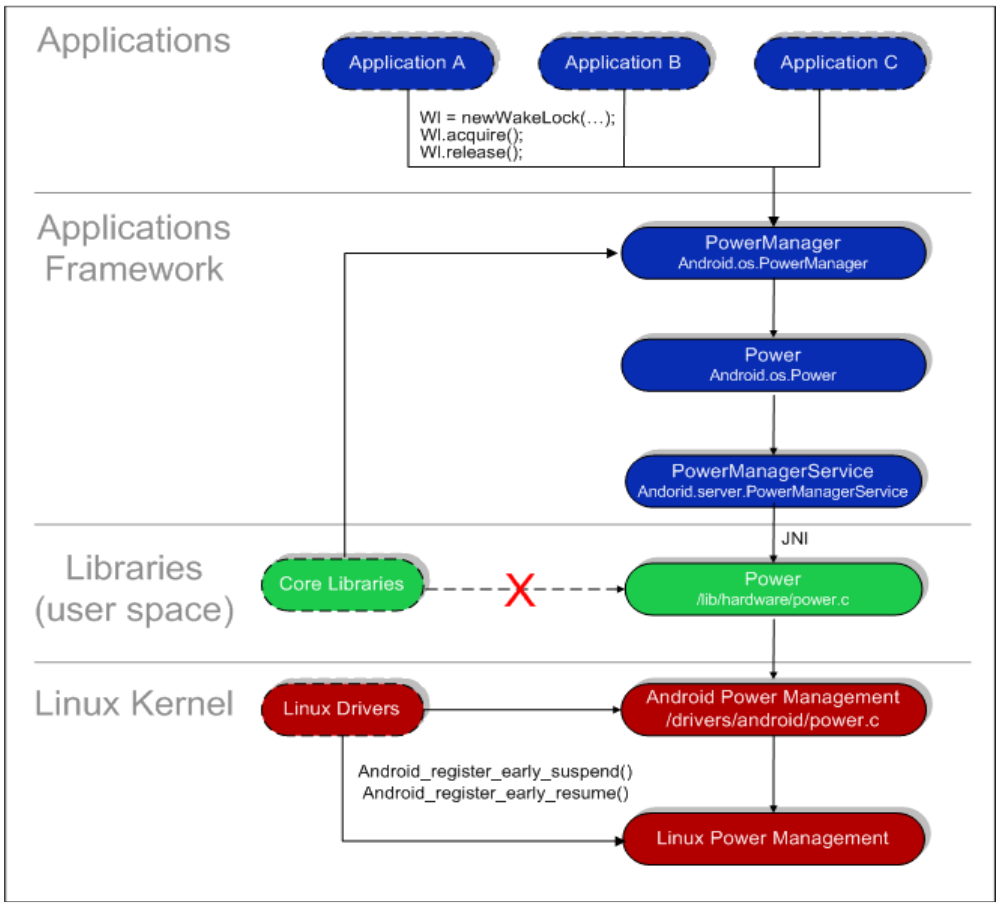

Figura 7. Arquitectura de administración de energía Android [11]

En la anterior imagen se muestra la arquitectura de administración de energía en Android. Esta arquitectura implementa un marco de aplicación sobre el kernel denominado *Power Management Applications Framework*. A través de este marco de aplicación, las aplicaciones de espacio de usuario pueden acceder a la clase *PowerManager* para controlar el estado de consumo en el que se encuentra el dispositivo. El módulo *Power* ofrece los drivers de bajo nivel que controlan los periféricos soportados por la clase *PowerManager*. El control se tiene sobre la visualización de la pantalla, y la iluminación de la pantalla, el teclado y los botones. El consumo de cada periférico se controla mediante el uso de los wakelocks, que se solicitan mediante la API de Android cuando una aplicación necesita que uno de los periféricos permanezca encendido. Si no hay ningún wakelock activo, entonces el dispositivo se suspende para ahorrar energía. A continuación se detallan los wakelocks que existen en Android:

- FULL_WAKE_LOCK
  Wakelock que mantiene la pantalla y el teclado encendidos con el máximo brillo.

- SCREEN_BRIGHT_WAKE_LOCK
  Wakelockque asegura que la pantalla está encendida con el máximo brillo, mientras que el teclado puede apagar su iluminación.

- SCREEN_DIM_WAKE_LOCK
  Wakelock que mantiene la pantalla encendida aunque permite bajar el brillo y el teclado puede apagar su iluminación.

- PARTIAL_WAKE_LOCK
  Wakelock que asegura que la CPU se mantiene activa aunque la pantalla puede apagarse.

Tabla 1. Resumen de los tipos de wakelocks de Android

FLAG VALUE	CPU	SCREEN	KEYBOARD
PARTIAL_WAKE_LOCK	ON	OFF	OFF
SCREEN_DIM_WAKE_LOCK	ON	DIM	OFF
SCREEN_BRIGHT_WAKE_LOCK	ON	BRIGHT	OFF
FULL_WAKE_LOCK	ON	BRIGHT	BRIGHT

Existen dos wakelocks más, que no están destinados al control de periféricos, y ejercen otras funciones de control sobre el dispositivo. Son los siguientes:

- ON_AFTER_RELEASE
  Cuando este wakelock se libera, el temporizador de actividad del usuario se resetea, de forma que la pantalla permanece encendida un periodo un poco más largo.

- ACQUIRE_CAUSES_WAKEUP
  Los anteriores wakelocks se utilizan para mantener el dispositivo activo cuando ya se encuentra activo. Sin embargo, este wakelock se utiliza cuando se quiere despertar al dispositivo cuando se encuentra suspendido. Normalmente se utiliza para realizar notificaciones**.**

En la siguiente figura se muestra la arquitectura del mecanismo de gestión de energía de Android mediante wakelocks:

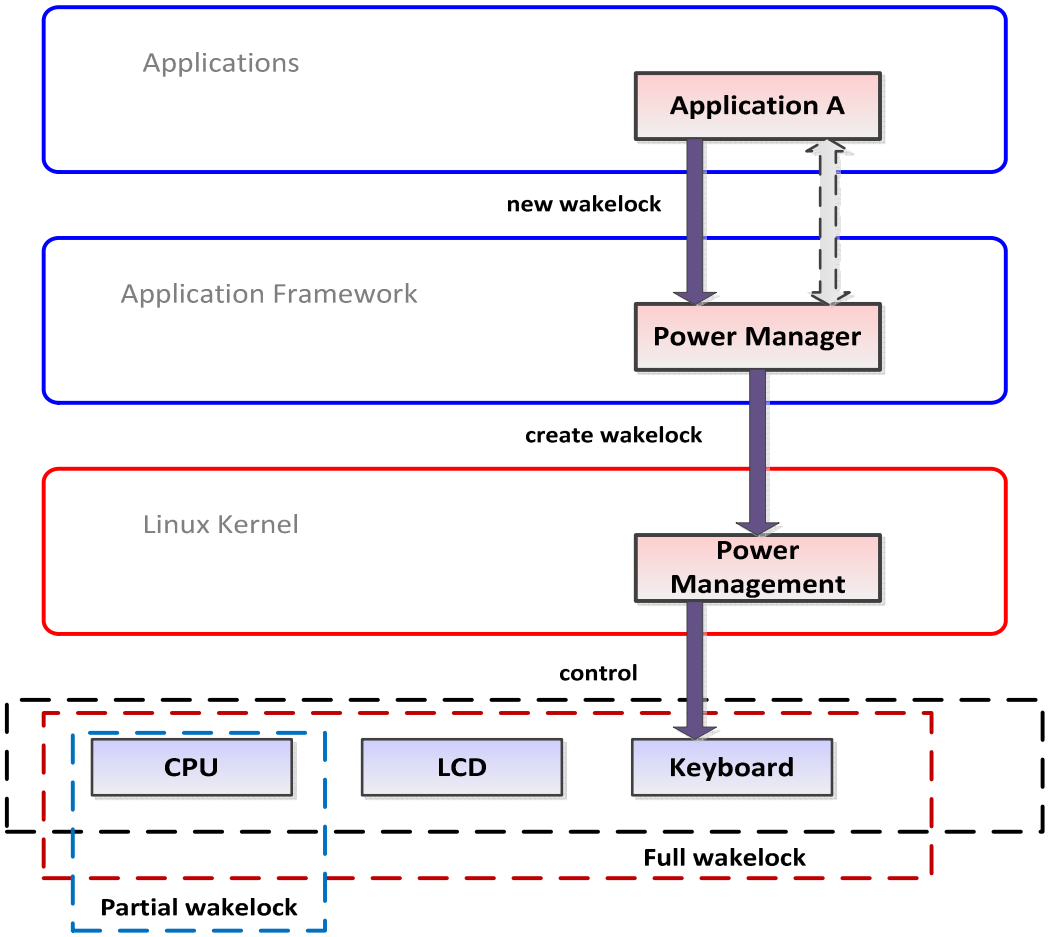

Figura 8. Arquitectura de administración de energía mediante wakelocks [12]

Tal y como se puede ver en la figura anterior, las aplicaciones y servicios solicitan recursos mediante wakelocks a través del marco de aplicación y las librerías nativas de Linux. Cuando se obtiene un wakelock, por ejemplo el FULL_WAKE_LOCK, se evita que la pantalla y el teclado se apaguen, y la CPU permanece activa. Este estado se mantiene hasta que el wakelock sea liberado. Por esta razón, los desarrolladores deben elegir los wakelocks más apropiados para sus aplicaciones y mantenerlos activos solo durante el tiempo necesario.

Según el mecanismo de administración de energía, el flujo adecuado para utilizar un wakelock correctamente se muestra en la Figura 9.

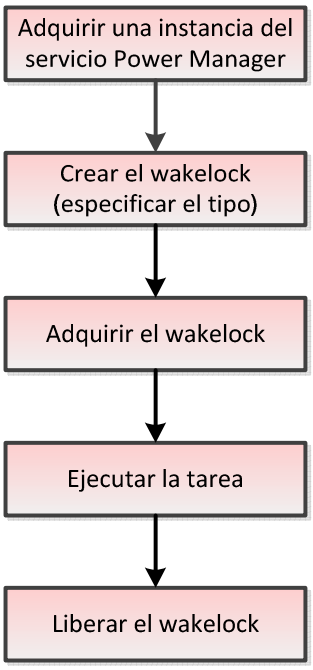

Figura 9. Flujo de vida de un wakelock

El uso de wakelocks permite ahorrar energía ya que si no hay ningún wakelock activo, el dispositivo pasa un estado de bajo consumo, que permite alargar la vida de la batería. Además de los wakelocks, el servicio PowerManager monitoriza el estado de la batería y el estado del dispositivo. Este servicio se coordina con la circuitería que controla la carga de la batería y apaga el terminal cuando se alcanza un nivel crítico.

2.1.4 Android Scripting

En general, el desarrollo de aplicaciones móviles en Android se realiza con el lenguaje de programación Java haciendo uso del SDK. Sin embargo, Android también permite programar y desarrollar programas y aplicaciones sencillas mediante scripts. Para ello, se ha creado el proyecto denominado SL4A (Scripting Layer for Android) [13].

El proyecto SL4A, que originalmente se denominaba ASE (Android Scriptig Environment), es, al igual que Android, un proyecto de código abierto de Google, distribuido bajo la licencia Apache 2.0. SL4A es una herramienta que permite desarrollar, ejecutar scripts e interactuar con intérpretes directamente en el propio dispositivo. SL4A incluye distintos lenguajes de scripting que tienen acceso a las APIs de Android mediante llamadas remotas (RPCs) a un servidor implementado como una aplicación Java estándar de Android [14].

Los lenguajes soportados por SL4A por ahora son: BeanShell, JRuby, Lua, PHP, Perl, Phyton y Rhino.

En su nivel más bajo, SL4A es un motor y entorno de ejecución de scripts, es decir, es una aplicación que contiene diferentes intérpretes, cada uno de ellos procesando un lenguaje específico. El repositorio del código fuente de SL4A está estructurado como un árbol jerárquico con cada tipo de lenguaje. Para transferir este código fuente a la plataforma Android, se realiza una compilación cruzada para la arquitectura ARM usando el NDK (Android Native Development Kit). El módulo Android se cargará como una biblioteca cuando SL4A lanza un intérprete específico y a partir de ese punto, los scripts serán interpretados línea a línea [15].

La arquitectura básica de SL4A es similar a la de un entorno de computación distribuido. En la Figura 10, se puede ver el flujo de ejecución cuando se lanza SL4A y a continuación se ejecuta un script. Cada script en SL4A debe ser un fichero externo, como hello.py escrito en Python, que define un número de funciones proxy necesarias para comunicarse con la API de Android.

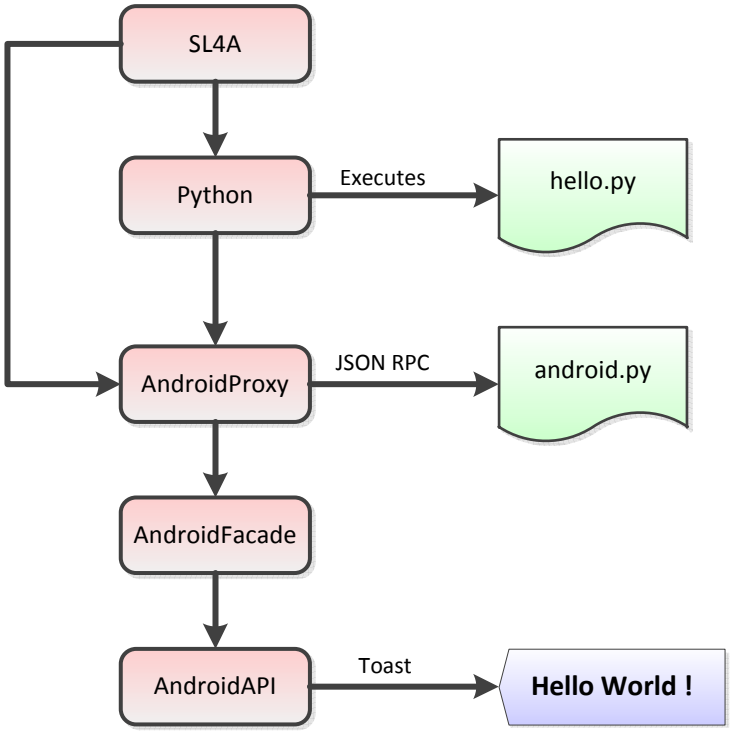

Figura 10. Diagrama de Flujo de ejecución de un script en SL4A [15]

La comunicación entre SL4A y el sistema operativo Android subyacente utiliza el procedimiento de llamada remoto, denominado RPC (Remote Procedure Call) y el

formato de datos JSON (JavaScript Object Notation). Normalmente RPC se utiliza en arquitecturas distribuidas en las que la información se transfiere de un cliente a un servidor. En el caso de SL4A, el servidor es el sistema operativo Android y el cliente es el script SL4A. Esto añade una capa de separación entre SL4A y Android, de forma que evita que scripts maliciosos causen algún perjuicio al sistema [15]. En la Figura 11, se muestra la arquitectura SL4A. El servidor RPC tiene acceso directo a la API de Android y se comporta como un *proxy* remoto utilizando la *Java Android API Facade* que encapsula y proporciona acceso a las APIs de Android. La *Java Android API Facade* es un servicio en un proceso separado y está implementada como un servidor de aplicación Android estándar en Java.

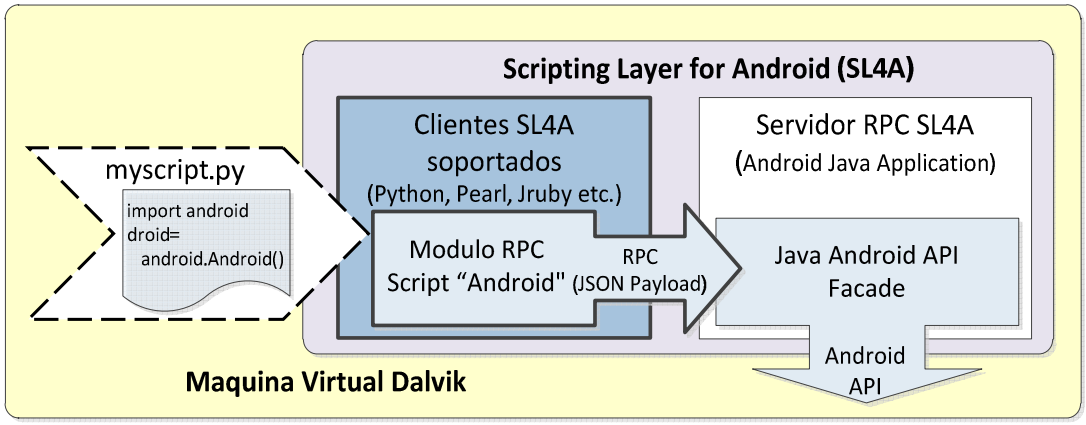

Figura 11. Arquitectura SL4A [14]

2.2 Consumo de energía en dispositivos móviles

Las nuevas funcionalidades técnicas de los dispositivos móviles han aumentado la demanda de una fuente de energía con mayor capacidad. Sin embargo, el progreso de la tecnología de las baterías no ha podido seguir el ritmo de la evolución de los terminales móviles. Los fabricantes de dispositivos móviles asumiendo que no es posible conseguir baterías de mayor capacidad, están dirigiendo sus esfuerzos en diseñar dispositivos móviles que consuman menos energía manteniendo sus mismas funcionalidades. Por otro lado, los desarrolladores de sistemas operativos móviles se han centrado en realizar una administración de energía más eficiente que alargue la duración de las baterías.

Cuando nos referimos al consumo de un dispositivo, se habla tanto del consumo de energía como de potencia. Con energía nos referimos a la capacidad para realizar un trabajo que se mide en Julios según el Sistema Internacional de Unidades.

Sin embargo, a la hora de realizar medidas normalmente el consumo en dispositivos electrónicos se suele medir en potencia. La potencia se define como la cantidad de trabajo realizado en un periodo de tiempo y se mide en vatios (1 W = 1 Julio/s). Un vatio se define como la tasa a la que se trabaja, cuando 1 Amperio de corriente atraviesa una diferencia de potencial de 1 Voltio. La potencia expresa la velocidad a la que se consume la energía en un dispositivo.

Potencia (W) = Energía (J)/Tiempo(s) = Voltaje (V) x Corriente (A)

2.2.1 Baterías en dispositivos móviles

La principal fuente de energía en un dispositivo móvil es una batería. La duración de la batería es una variable importante en el rendimiento del sistema. Las baterías de Níquel y Litio son las más comunes en terminales móviles. Los smartphones utilizan en general baterías de iones de Litio, que se denominan Li-ion.

Una batería puede caracterizarse mediante los siguientes parámetros:

- Densidad de energía (en unidades de potencia por unidad de peso, por ejemplo mA por Kg)
- Número de ciclos de carga que la batería puede soportar durante su tiempo de vida.
- Características de velocidad de carga
- Características de velocidad de descarga

Las baterías de Li-ion, que es la que tiene el dispositivo que se ha usado para la realización de este proyecto, son pequeñas, más ligeras y duraderas que otros tipos de batería, pero también son más caras. Además, son seguras para los usuarios y no sufren sobrecalentamientos durante la carga y la descarga. Las baterías de Li-ion tienen un bajo ciclo propio de descarga lo que las hace ideales para los dispositivos móviles, es decir, la corriente descarga cuando no se requiere energía es muy baja. En la Figura 12, se puede ver la característica de descarga de una batería Li-ion de grafito o coque, donde la capacidad (%) se refiere al nivel de batería descargada.

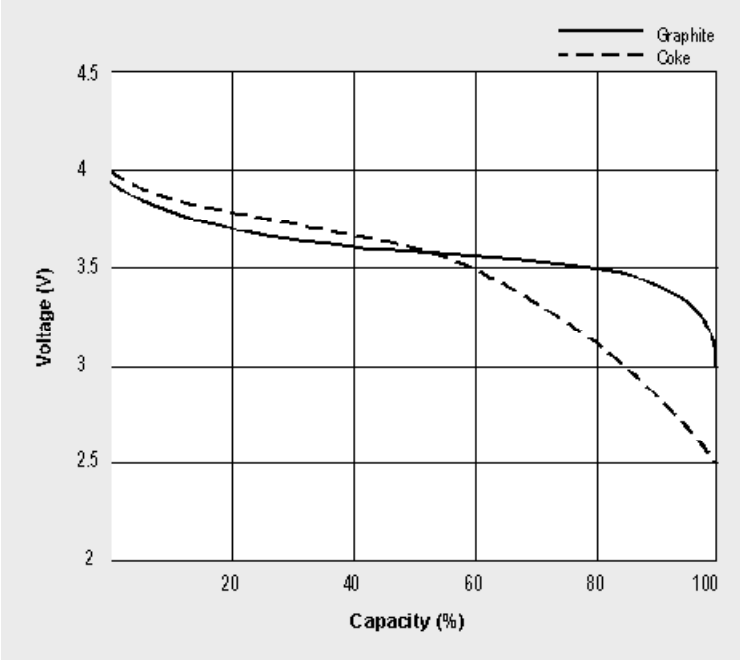

Figura 12. Característica de la descarga de una batería de Li-on [16]

La capacidad de ls baterías se expresa en mAh. Los smartphones suelen tener baterías con capacidad entre 1150 mAh y 1600 mAh. Conociendo la capacidad de la batería y la potencia consumida, en teoría, se podría obtener la velocidad de descarga y el tiempo restante de vida de la batería. Sin embargo, los dispositivos móviles se componen de un conjunto heterogéneo de componentes que se encuentran activos en diferentes momentos y con distintos niveles de actividad, consumiendo diferentes fracciones de potencia a lo largo de tiempo. Esto hace que sea difícil calcular la duración restante de una batería.

2.2.2 Caracterización del consumo de energía en un smartphone

En primer lugar, para evaluar potencia consumida por un dispositivo móvil hay que tener en cuenta que el dispositivo puede encontrarse en diferentes estados de consumo de potencia según el nivel de actividad. En general, un smartphone puede estar en uno de los siguientes estados:

- **Suspendido**: En este estado el dispositivo no está activamente siendo usado, el procesador está inactivo, y simplemente el subsistema de comunicaciones mantiene una pequeña actividad para permanecer conectado a la red de forma que se puedan recibir llamadas, mensajes e emails.

- **Inactivo**: El dispositivo se encuentra despierto pero ninguna aplicación está activa y la pantalla está apagada.

- **Activo:** Este es el estado en el que se encuentra el dispositivo cuando está ejecutando una tarea, una llamada, transmitiendo o recibiendo datos, o el usuario está interactuando con alguna aplicación.

Por otro lado, existen distintas formas para evaluar y medir la potencia consumida. Cada uno de estos métodos tiene sus ventajas y desventajas y es más apropiado dependiendo de la situación. Los métodos más habituales para obtener la potencia consumida en un dispositivo son los siguientes:

- **Medida de la potencia a nivel de componente**

Este método tiene en cuenta que un dispositivo móvil se puede ver como un conjunto heterogéneo de diferentes componentes. La medida de la potencia se realiza para un componente en particular, como por ejemplo la pantalla o la CPU, y la potencia total medida es la suma de todas las medidas de cada uno de los componentes importantes. Los componentes importantes son aquellos que influyen en nivel de potencia total consumida por el dispositivo. Esta metodología se caracteriza por:

- Como las medidas se realizan para cada componente independientemente, los resultados son más precisos y las medidas se pueden reproducir fácilmente.

- Es más complicado, conlleva más esfuerzo, tiempo y mayor coste.

- Se necesita conocer detalladamente el software y el hardware del dispositivo para encontrar los componentes y la forma de evaluarlos correctamente.

- **Medida de la potencia a nivel de dispositivo**

Con esta metodología la potencia del dispositivo se mide de forma global para varios escenarios de uso. Medir la potencia a nivel de dispositivo es la forma más habitual para caracterizar el consumo y evaluar la vida de la batería en smartphones. Las características principales de este método son:

- Es sencillo y más práctico para la mayoría de operadores y desarrolladores de aplicaciones.

- Dependiendo de lo preciso que sea el dispositivo a evaluar, los resultados pueden variar de un test a otro, por lo que puede ser necesario repetir las medidas hasta conseguir estadística y estabilidad.

La Figura 13 muestra los diferentes componentes de un típico smartphone. A la hora evaluar cada uno de estos componentes es importante definir la carga y el estado en el que se va a encontrar cada uno de ellos en el momento de las medidas para estas sean fiables y precisas. Entre todos los componentes, los que requieren mayor potencia para su funcionamiento suelen ser las tecnologías de comunicación. La pantalla es otro

componente con un alto consumo, ya que cada vez las pantallas son más grandes y tienen mayor calidad.

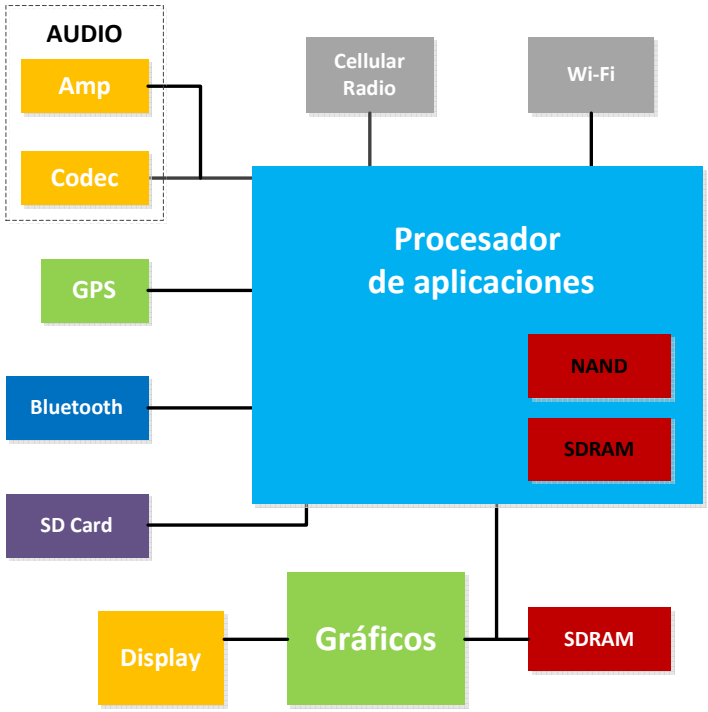

Figura 13. Principales componentes en un smartphone [16]

2.3 Tecnologías de comunicación

En esta sección se presentan dos de las tecnologías más utilizadas para conexiones de datos en terminales móviles: Wi-Fi y 3G.

En Noviembre 2011, el número mundial de usuarios de conexiones de banda ancha en un dispositivo móvil era de más de 1 billón [17], estando el 45% de la población mundial bajo la cobertura de una red 3G [18]. Por otro lado, según la WiFi-Alliance [19] hay 700 millones de usuarios Wi-Fi y cada año 800 millones de nuevos dispositivos Wi-Fi.

Actualmente, la mayoría de smartphones permiten la conexión a redes Wi-Fi y 3G. Sin embargo, estas tecnologías son muy diferentes. Ambas soportan datos y voz, pero una red Wi-Fi está destinada a conectar clientes en un área geográfica pequeña, mientras que una red 3G provee conectividad en un área muy amplia.

2.3.1 Wi-Fi

Wi-Fi [20] es una tecnología que permite conectar dispositivos electrónicos entre sí, a Internet y a redes cableadas que usan tecnología Ethernet. La Wi-Fi Alliance [19] define como Wi-Fi, a la certificación otorgada a cualquier dispositivo que cumple con la familia estándares IEEE 802.11.

Las especificaciones IEEE 802.11 [22] definen las características de los dos niveles inferiores de la arquitectura OSI (nivel físico y de enlace de datos), en una red de área local inalámbrica (WLAN).

- La capa física (**PHY**) define la modulación de las ondas de radio y las características de señalización para la transmisión de datos. Se especifican tres tecnologías en el medio físico: Espectro Ensanchado por Secuencia Directa (DSSS), Espectro Ensanchado por Salto en Frecuencia (FHSS) e Infrarrojos.

- La capa de enlace de datos compuesta por dos subcapas: Control de Enlace Lógico (**LLC)** y Control de Acceso al Medio (**MAC**), que definen la interfaz con la capa física, y las reglas para la comunicación entre las estaciones de la red. Como protocolo de acceso al medio se utiliza CSMA/CA (Carrier Sense Multiple Access / Collision Advoidance).

Una red inalámbrica Wi-Fi funciona en términos generales de forma similar a una red Ethernet (IEEE 802.3), de hecho cualquier protocolo de nivel superior puede utilizarse de la misma manera que en una red Ethernet. La única diferencia es que en lugar de cableado, la tecnología Wi-Fi utiliza la radiofrecuencia como medio de transmisión.

Una red Wi-Fi opera en la banda de frecuencia de 2.4 GHz o en la banda de 5 GHz. Estas bandas de frecuencias no requieren licencia para su uso, por lo que los proveedores de este tipo de redes no tienen que pagar a los gobiernos por utilizarlas. Sin embargo, hay muchas más interferencias con otros usuarios siendo más complejo operar en ellas.

Atendiendo a la frecuencia de operación y a las velocidades posibles de transmisión, se definen distintos tipos de tecnologías Wi-Fi, los más usados se indican a continuación.

Tabla 2. Resumen de las principales tecnologías Wi-Fi definidas por el estándar IEEE802.11

Tecnología Wi-Fi	Banda de Frecuencias	Máximo ancho de banda o velocidad de transmisión
802.11a	5 GHz	54 Mbps
802.11b	2.4 GHz	11 Mbps
802.11g	2.4 GHz	54 Mbps
802.11n	2.4 GHz 5 GHz	450 Mbps

El estándar 802.11 define dos tipos de arquitectura de red:

- **Infraestructura**: Es la arquitectura de res más habitual. Esta arquitectura está formada por distintos tipos de entidades físicas. En primer lugar, siempre hay al menos una estación (STA) con una interfaz de conexión a la red Wi-Fi, como puede ser un ordenador o un teléfono. Por otro lado, debe haber un punto de acceso (AP) o varios que conecten la red inalámbrica WLAN con la red cableada LAN. El número de puntos de acceso dependerá del área de cobertura que se quiera tener. El área de cobertura ofrecida por un AP se denomina área de servicio básico (BSA) y al conjunto formado por el punto de acceso más las estaciones dentro su BSA, conjunto de servicio básico (BSS). Por último, se encuentra el sistema de distribución (DS) que es la red cableada con la que se comunica el AP y que permite la comunicación con otros BSS y otras redes. Es el tipo de arquitectura de red más habitual.

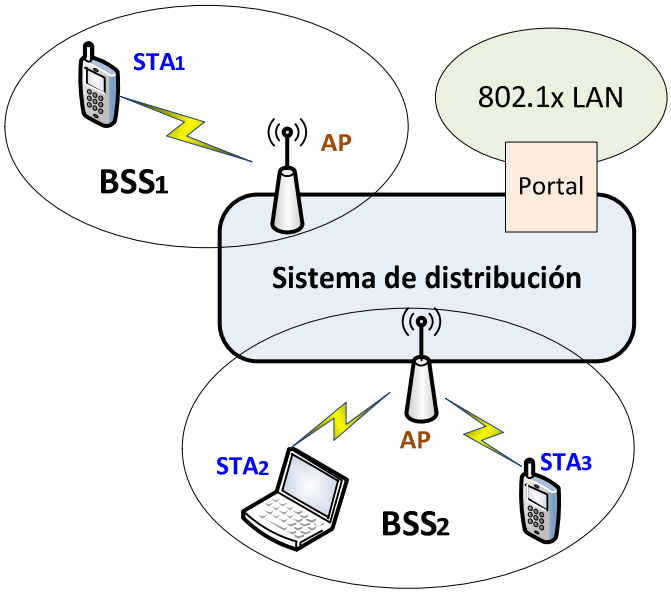

Figura 14. Arquitectura de red infraestructura

- **Ad-hoc**: Las estaciones se comunican directamente con otras estaciones, de tal forma que se encuentran dentro de un mismo radio de cobertura y no se usan puntos de acceso. Este tipo de arquitectura se usa en redes con un número pequeño de estaciones.

Una estación para poder transmitir o recibir datos a través de un AP en primer lugar debe establecer una conexión al mismo. En la Figura 15, se muestra el proceso básico de establecimiento de conexión en 802.11, que se realiza a nivel de enlace. Podemos resumir este proceso en tres pasos: escaneo, autenticación y asociación.

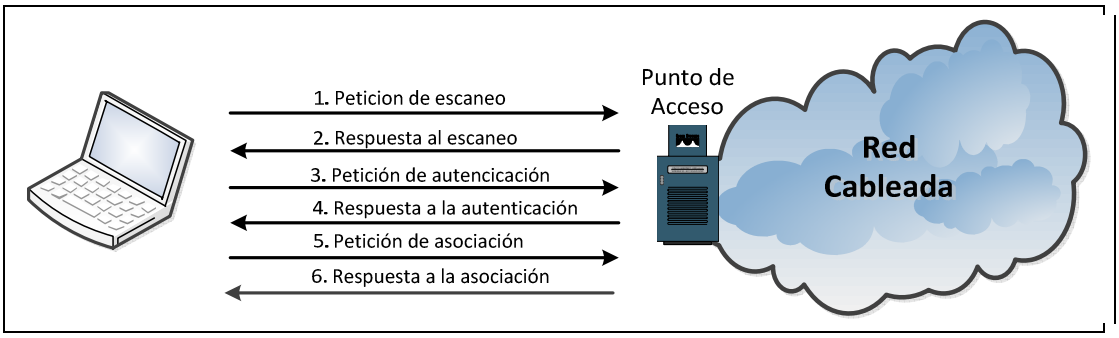

Figura 15. Proceso para establecer una conexión entre una STA y un AP [21]

En primer lugar, una estación que se quiere conectar a una red Wi-Fi tiene encontrar una red mediante un proceso denominado *Escaneo* que puede ser en modo pasivo o activo. En modo pasivo la estación espera a escuchar lo que se denomina *Beacons*, mensajes que emiten los APs para anunciar sus redes. En el modo activo las estaciones envían mensajes de prueba en todo los canales para obtener respuesta de los APs disponibles. Al final del proceso de escaneo, la estación tiene una lista con todas las redes disponibles.

A continuación, una vez que la estación escoge a que red se va a conectar, el AP es quien autoriza la conexión a la misma. El estándar 802.11 especifica dos tipos de autenticación: abierta (OSA) o de clave compartida (SKA).

El último paso es la asociación cuyo objetivo es establecer una conexión lógica entre la estación y el AP. El AP después de aceptar la petición de asociación, registra e identifica la conexión Una vez que la estación está asociada con el AP ya puede enviar y recibir datos, ya que el sistema de distribución conoce que existe un nodo móvil al que el AP está dando servicio.

2.3.1.1 Administración de energía en IEEE 802.11

Ya que la mayoría de los dispositivos que se conectan a una red Wi-Fi tienen baterías de capacidad limitada, la administración de potencia es muy importante. El estándar IEEE 802.11 especifica el algoritmo PSM (*Power Saving Mode*) que permite a los dispositivos administrar dinámicamente la potencia consumida por la tarjeta de la interfaz inalámbrica, que denomina WNIC (Wireless Network Interface Card). Según el protocolo PSM, esta tarjeta puede encontrarse en cinco estados de consumo:

- Apagada (*Power-off*): En este estado la tarjeta WNIC no consume energía.

- Suspendida (*Sleep*): La tarjeta WNIC está encendida pero no asociada con ningún AP, por lo que no se puede transmitir ni recibir. El consumo es menor que cuando la tarjeta está inactiva.

- Inactiva (*Idle)*: La tarjeta WNIC está encendida y preparada para transmitir o recibir. Consume una cantidad significante de energía.

- Activa (*Transmit/Receive*): La tarjeta WNIC transmite o recibe datos, por lo que el consumo en este estado es el más elevado.

En general, el comportamiento en el protocolo PSM de los dispositivos de los usuarios y los APs se resume a continuación:

- Puntos de acceso: Mientras la tarjeta WNIC de un usuario está suspendida, lo paquetes de datos son almacenados por AP. El AP periódicamente transmite las tramas *Beacon* que contienen el indicador de tráfico TIM con los dispositivos que tienen al menos un paquete almacenado en el AP. El AP envía lo datos que tiene almacenados al usuario después de que el dispositivo envíe la petición en una trama *PS-Poll*.

- Tarjetas WNIC: Si el protocolo PSM está funcionando en una tarjeta, esta se suspende durante un periodo fijo de tiempo al expirar un temporizador desde la última vez que se envió un paquete. Después del periodo de suspensión, la tarjeta vuelve a activarse para escuchar tramas *Beacon*. Por lo tanto, los dispositivos están sincronizados con los APs. Si el dispositivo aparece en el indicador TIM de la trama Beacon, envía al AP un *PS-Poll* solicitando que le envíe los datos que tiene almacenados.

El coste inicial de asociarse a un AP es alto. Sin embargo, ya que la mayoría de dispositivos utilizan PSM, una vez asociado, el coste de mantener la asociación es pequeño. Pero la mayor parte de energía se consume en las transmisiones de datos, siendo la energía consumida proporcional al tamaño de los datos transferidos.

2.3.2 3G

3G [23] es la abreviación de la tecnología móvil de tercera generación para teléfonos móviles y servicios de telecomunicaciones que cumplen con el estándar IMT-2000 definido por la Unión Internacional de Telecomunicaciones (ITU).

La tecnología 3G aparece como respuesta a la necesidad de un ancho de banda mayor en redes celulares, ya que los estándares de segunda generación (2G) no fueron diseñados para la transmisión de datos en modo paquete. El fin de 3G es proporcionar un servicio orientado a paquetes a distancia para transmitir video, texto y voz digital. En general, soporta tasas de entre 384 Kbps en un vehículo en movimiento y hasta 2Mbps para usuarios sin movimiento o andando.

Dentro de la especificación IMT-2000 se definen los estándares UMTS y CDMA2000. CDMA2000 es el estándar que se usa especialmente en Norte América y Corea del Sur. Como su propio nombre indica utiliza el esquema de acceso múltiple CDMA. UMTS es el estándar para Europa, y por tanto el que se ha utilizado para este proyecto.

UMTS [24] o también denominado **Sistema Universal de Telecomunicaciones Móviles** fue desarrollado por 3GPP[3] y es el estándar para Europa, Japón y China. UMTS ofrece varias opciones para la interfaz radio, denominada UTRA (Acceso Radio Terrestre UMTS), que comparten la misma infraestructura de red:

- **W-CDMA** (Acceso múltiple por división de código de banda ancha): Aumenta las tasas de transmisión de datos de los sistemas GSM utilizando la interfaz CDMA (Acceso múltiple por división de código) en lugar de TDMA (Acceso Múltiple por División de Tiempo). CDMA es una tecnología de acceso que habilita múltiples usuarios de telefonía simultáneamente hacia una estación base mientras sus conversaciones transcurren de forma independiente. Utiliza los modos de operación FDD (División en Frecuencia Dúplex) y TDD (División en Tiempo Duplex) para permitir a los usuarios beneficios a la hora de repartir el espectro utilizado.

- **TD-SCDMA** (Tecnología CDMA síncrona por división en el tiempo): Utiliza el modo TDD y se utiliza solo en China.

- **HSPA+**: Es la última en aparecer y puede proporcionar tasas de datos de hasta 56 Mbit/s de bajada y 22 Mbit/s de subida gracias a una técnica multi-antena conocida como MIMO (Multiple-Input Multiple-Output) y la modulación 64-QAM.

En UMTS el espectro de frecuencias funciona bajo licencias. Las bandas de frecuencias 1885-2025 MHz y 2110-2200 MHz están reservadas para los sistemas que cumplen con la especificación IMT-2000 y las bandas 1980-2010 MHz y 2170-2200 MHz para la parte satélite de estos sistemas.

UMTS está construida sobre la infraestructura ya existente de GSM por lo que permite utilizar en paralelo GSM con UMTS. La integración de estas dos redes ha permitido una transición suave hacia UMTS. Este funcionamiento en paralelo es posible gracias a que UMTS utiliza bandas de frecuencias distintas a las de GSM. De esta forma con UMTS se pueden utilizar servicios multimedia y aplicaciones web mientras que el usuario habla por teléfono al mismo tiempo.

Aunque la arquitectura de la red UMTS depende del proveedor de red, en general se compone de los siguientes elementos principales:

- **Núcleo de Red (Core Network)**: Incorpora las funciones de transporte e inteligencia de la red. Por un lado, soporta el transporte de la información de tráfico y señalización. Por otro lado, las funciones de inteligencia son llevadas a cabo por el encaminamiento. Además, el núcleo de red permite la conexión con

[3] 3GPP es una colaboración de grupos de asociaciones de telecomunicaciones que se formó con el objetivo de establecer las especificaciones de un sistema global de comunicaciones de tercera generación para móviles (UMTS) basándose en las especificaciones de GSM dentro del marco del proyecto internacional IMT-2000.

otras redes de telecomunicaciones, de forma que resulte posible la comunicación con usuarios que se encuentren en otras redes distintas a UMTS.

- **Red de Acceso Radio (UTRAN):** Es la red de acceso que mediante dos interfaces conecta con el núcleo de red y con los equipos de usuario, la interfaz I_u y la interfaz U_u, respectivamente. Se compone de una serie de sistemas de Control de Red Radio (RNC) y de un conjunto de estaciones base, también denominadas Nodos B.

- **Equipos de Usuario (UE):** Son los dispositivos móviles que los usuarios utilizan para iniciar la comunicación, conectándose a las estaciones base cuando existe cobertura. Deben estar preparados para soportar el estándar y los protocolos para los que fue diseñado, es decir, tiene que ser capaz de acceder a la red UTRAN mediante la tecnología W-CDMA, para establecer la comunicación con otro UE o con otros dispositivos de otras redes.

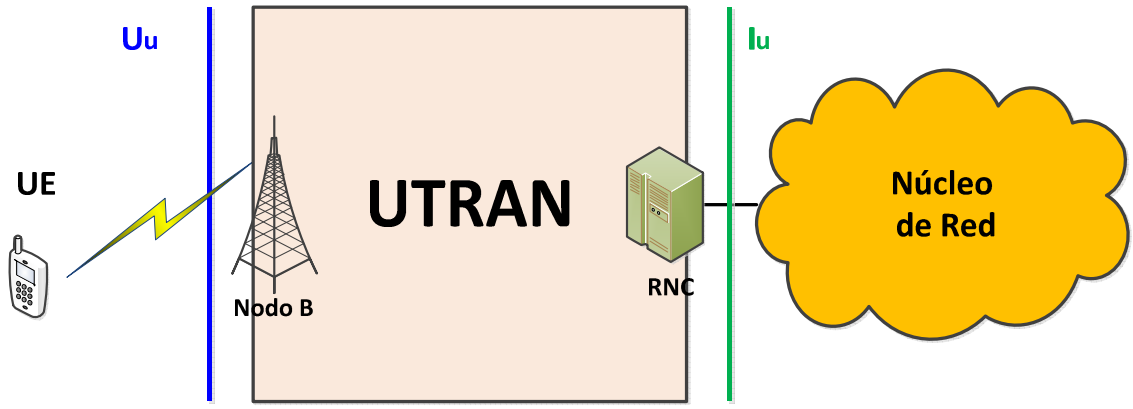

Figura 16. Arquitectura de una red UMTS

2.3.2.1 Administración de energía en 3G

Dos factores determinan el consumo de energía debido a la actividad de la red 3G en un dispositivo móvil [3]. El primer factor es la transmisión de energía proporcional a la longitud de la transmisión y el nivel de potencia transmitida. El segundo factor, es el protocolo RCC (Radio Resource Control) [25] que es responsable de la asignación de los recursos radio, lo cual afecta a la energía consumida por el dispositivo y la experiencia del usuario.

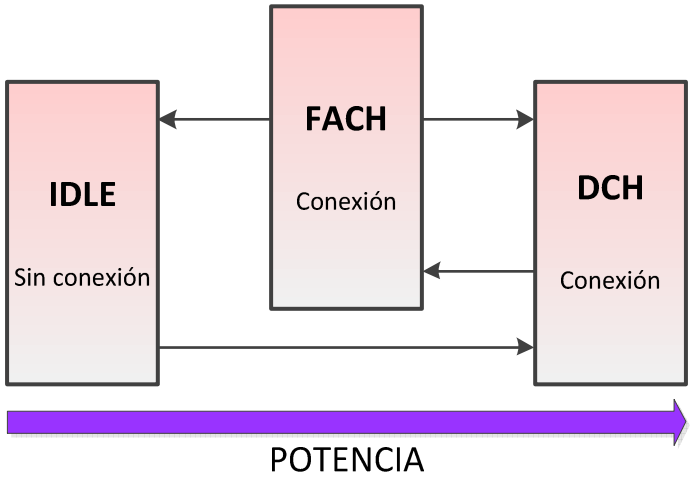

Figura 17. Máquina de estados RCC para una red 3G[3]

En la Figura 17, se muestra la máquina de estados del protocolo RCC para una red UMTS/WCDMA (3G) según el estándar 3GPP. Hay tres estados en los que puede encontrarse el UE, en cada uno de los cuales los recursos asignados son diferentes.

- **IDLE:** En este estado no hay conexión establecida, ni ningún recurso asignado, siendo no posible la transferencia de datos. Por lo tanto, es el estado en el que se consume menos energía y la transición a otro estado, supone moverse a un estado de mayor consumo de energía.

- **DCH (Dedicated Channel):**Se establece la conexión, reservando un canal dedicado al UE y con un gran ancho de banda y un retraso pequeño en las conexiones, pero con el coste de un alto consumo de energía.

- **FACH (Forward Access Channel):** En este estado existe conexión pero no hay un canal dedicado para el UE, sino que se comparte el canal con otros dispositivos cuando el tráfico que se transmite es pequeño.

En el estado IDLE, la energía consumida por la interfaz radio del UE es casi despreciable, mientras que en los estados DCH y FACH el consumo se incrementa considerablemente. La potencia consumida en el estado DCH es de un 50% a un 100% mayor que en el estado FCH. Mientras nos encontremos en un mismo estado, la energía consumida se mantiene constante, independientemente del ancho de banda, para un mismo nivel de señal recibida [26].

La transición entre estados se debe a los temporizadores de inactividad. La transición de un estado de alto consumo a uno de menor consumo ocurre cuando la conexión 3G ha estado inactiva durante un periodo igual al temporizador de inactividad. Este mecanismo permite un gran ahorro de energía ya que evita que los dispositivos se encuentren en estados de alto consumo cuando no es necesario.

Los temporizadores de inactividad los establecen los operadores, por lo que puede variar de unos a otros. Su valor es crítico, ya que por un lado, si la transición entre estados es muy frecuente supone mayores retrasos en el UE y tiempos de procesado más largos en la red UTRAN. Por otro lado, si el tiempo de inactividad permitido es largo, estaremos en estados de alto consumo durante más tiempo sin ser necesario, siendo la energía consumida mayor.

Capítulo 3

Metodología empleada

Este capítulo, está dedicado a la metodología que se ha desarrollado para llevar a cabo el estudio del consumo de energía en un smartphone Android. Cuando se habla de medir el consumo de energía de un dispositivo, lo más intuitivo es pensar en utilizar aparatos electrónicos de medida. Sin embargo, viendo las posibilidades que ofrece la plataforma Android se ha buscado una metodología que se pueda llevar a cabo en el propio dispositivo sin necesidad de realizar complicados montajes ni utilizar hardware adicional. Para ello se plantean una metodología según los siguientes objetivos:

- Obtener la potencia consumida de un dispositivo Android de manera sencilla.
- Caracterizar el consumo de los distintos componentes del dispositivo.
- Concluir cuál de las tecnologías de comunicación estudiadas es más eficiente en cuanto a consumo de energía.

Para presentar la metodología que cumple los anteriores objetivos, en primer lugar se detallan los dispositivos y herramientas que se han utilizado en el desarrollo de este proyecto. Además, se explica y justifica cómo se consigue obtener la potencia consumida en un dispositivo Android aprovechando las facilidades que ofrece el SDK para conocer el estado de la batería, sin necesidad de utilizar ningún aparato electrónico de medida. De esta forma, se describe como se ha realizado una aplicación que monitoriza el estado de la batería del teléfono de la forma más sencilla y eficiente posible.

Por último, se va a detallar la manera en la que se ha obtenido la potencia consumida por cada uno de los componentes que se han considerado importantes para el estudio del consumo de energía en un smartphone. Se describe como se realizan las medidas y los diferentes escenarios que se han planteado según la tecnología de comunicación evaluada.

3.1 Entorno de medida

3.1.1 Dispositivos

El estudio del consumo en un dispositivo con sistema operativo Android se ha llevado a cabo mediante el smartphone HTC Legend con Android 2.1 Eclair, que opera sobre la versión 2.6.29-9a3026a7 del kernel de Linux. Los principales componentes del dispositivo se detallan en la Tabla 3. Es necesario tener acceso *root*[4] al HTC Legend para poder realizar algunas operaciones llevadas a cabo en el desarrollo del proyecto, que con el modo de acceso por defecto no se permiten. En el Anexo D, se explican los pasos necesarios para tener acceso root en este dispositivo.

Figura 18. HTC Legend

[4] Ser root o administrador de un dispositivo supone tener los máximos privilegios para operar sobre él.

Tabla 3. Principales componentes del HTC Legend [27]

Sistema Operativo	Android 2.1 (Éclair)
CPU	Qualcomm MSM 7227 @600MHz
Memoria	RAM: 384 MB ROM: 512 MB
Red	HSPA/WCDMA: ▪ Europa/Asia: 900/2100 MHz ▪ Velocidad de subida de hasta 2 Mbps y velocidad de bajada de 7,2 Mbps máximo Cuatro bandas GSM/GPRS/EDGE: ▪ 850/900/1800/1900 MHz
GPS	Antena GPS interna
Pantalla	AMOLED de 3,2 pulgadas con resolución HVGA 320 x 480
Sensores	Sensor-G Brújula digital Sensor de proximidad Sensor de luz ambiente
Interfaz Wi-Fi	IEEE 802.11b/g Texas Instruments WL1273 chipset
Interfaz Bluetooth	Bluetooth 2.1 con FTP/OPP
Batería	1300 mAh
Almacenamiento externo	Slot para tarjeta microSD

Por otro lado, el equipo empleado para el desarrollo y comunicación con el dispositivo Android, es un ordenador con sistema operativo Linux Ubuntu 10.4 con conexión a Internet. El equipo dispone de un procesador de doble núcleo @2.66GHz, disco duro de 280GB y 2GB de RAM. Además, este equipo dispone una interfaz inalámbrica Atheros con driver ath5k para IEEE a/b/g, que se configura como un punto de acceso para el escenario de medidas con la tecnología Wi-Fi.

Para la comunicación 3G se dispone de una tarjeta SIM, del proveedor de telefonía Orange.

3.1.2 Herramientas software

Además de los dispositivos físicos han sido necesarias una serie de herramientas software para llevar a cabo este proyecto. A continuación, se describe estas herramientas y como se han utilizado en el estudio del consumo de energía en un dispositivo Android.

➢ SDK de Android y entorno de desarrollo

El SDK (*Software Development Kit*) de Android es un kit de desarrollo, que contiene las librerías (API) y las herramientas necesarias para desarrollar aplicaciones en Android. Las herramientas del SDK se dividen en dos grupos: *SDK tools* and *Platfform tools*. Las *SDK tools* son necesarias e independientes de la versión de la plataforma de Android en la que se esté desarrollando. Las *Platfform tools* son actualizadas para soportar las nuevas características que van añadiendo las últimas versiones de la plataforma Android.

Para este proyecto se ha necesitado la API 7 que es la que soporta la versión Android 2.1, instalada en el HTC Legend. De las herramientas disponibles se hace uso de la herramienta *Android Debug Bridge* (ADB) que se describe en la siguiente sección.

Por otro lado, el entorno de desarrollo empleado ha sido el IDE de Eclipse junto los componentes que se detallan a continuación, necesarios para construir una aplicación Android:

- JDK 6 (kit de desarrollo de Java).

- Android Development Tools (ADT) plug-in.

ADT es un plug-in específico para el IDE de Eclipse, que permite utilizar las herramientas del SDK para construir aplicaciones Android. ADT extiende las capacidades de Eclipse de forma que se pueden crear de forma rápida y sencilla proyectos Android, aplicaciones con interfaz de usuario, depurar aplicaciones usando las herramientas del SDK, e incluso distribuir las aplicaciones creadas. Este plug-in no está incluido en el SDK, sino que su instalación se realiza desde Eclipse. Utilizar Eclipse con ADT es la opción más sencilla y recomendada para el desarrollo Android, pero si se quiere trabajar en otro IDE, no es necesario instalar Eclipse o ADT. En su lugar, se pueden utilizar directamente las herramientas del SDK de Android para construir y depurar aplicaciones, aunque es mucho más complejo.

La instalación de los componentes del SDK y el entorno de desarrollo se explica en el Anexo A.

➢ ADB

Android Debug Bridge (ADB) [28] es una herramienta de línea de comandos que permite la comunicación del equipo de desarrollo con el emulador o con los dispositivos físicos Android. Es un programa cliente-servidor que incluye tres componentes:

- Un *daemon* que se ejecuta como un proceso en segundo plano en el emulador o dispositivo Android.

- Un cliente que se ejecuta en el equipo de desarrollo. Se puede invocar un cliente desde un terminal mediante el comando adb.

- Un servidor que se ejecuta como un proceso en segundo plano en el equipo de desarrollo. El servidor gestiona la comunicación entre el cliente y el daemon adb que corre en el emulador o el dispositivo.

Cuando se inicia un cliente adb, este primero comprueba si hay un proceso servidor activo y si no lo hay lo crea. Cuando el servidor se inicia, se asocia al puerto TCP local 5037 y permanece escuchando comandos enviados por los clientes adb, que también usan el puerto 5037 para comunicarse con el servidor. A continuación, el servidor establece conexiones a todas las instancias de emuladores o dispositivos activos, que son localizados por el servidor mediante el escaneo de todos los puertos impares en el rango 5555-5585. En los puertos en los que el servidor encuentra un daemon adb, establece una conexión. Una vez que el servidor ha establecido la conexión con todas las instancias de emuladores y dispositivos, se puede utilizar los comandos adb para controlar y acceder a estas instancias.

Esta herramienta permite realizar operaciones, sobre el emulador o dispositivo Android desde un terminal de línea de comandos, tales como instalar aplicaciones, obtener una shell, copiar archivos y obtener información del dispositivo. Como configurar la herramienta ADB y utilizar las operaciones disponibles, se explica en el Anexo E.

- **tiwlan_cu**

El módulo *tiwlan_cu* es una interfaz de línea de comandos que permite configurar el driver de la interfaz IEEE 802.11. El acceso a este módulo se realiza a través de la herramienta anteriormente descrita ADB, ya que hay que acceder al teléfono con una shell. Con el módulo tiwlan_cu se pueden realizar operaciones como escanear redes, conectarse a una red, etc. y establecer diferentes configuraciones de roaming, escaneo, seguridad y administración de energía.

En este proyecto se ha utilizado este módulo en lugar del administrador de conexiones inalámbricas, para poder configurar el driver de la interfaz IEEE 802.11 y establecer una conexión con la red creada en el laboratorio de trabajo.

En el Anexo E, se ofrece información sobre cómo se ha utilizado este módulo.

- **SL4A**

Tal y como se explicó en la sección 2.1.4, la herramienta SL4A permite crear y ejecutar scripts que acceden a las librerías de Android en el propio terminal. En este proyecto, esta herramienta se ha utilizado para ejecutar los scripts que establecen las condiciones adecuadas para realizar las medidas del consumo de cada componente.

Los scripts se han desarrollado con el lenguaje Python. Python es un lenguaje de programación interpretado que combina potencia con una sintaxis limpia y sencilla. Soporta orientación a objetos, programación imperativa, y en menor medida,

programación funcional. Su elegante sintaxis, su gestión de tipos dinámica y su naturaleza interpretada hacen que sea un lenguaje ideal para el desarrollo de scripts y aplicaciones en muchas áreas y en la mayoría de plataformas.

En un terminal Android, para poder ejecutar código Python es necesario instalar el intérprete de Python para Android a través de la herramienta SL4A. En el Anexo C, se describen los pasos para realizar la instalación de SL4A y el intérprete de Python en un dispositivo Android.

Para acceder a las funciones de la API de Android disponibles desde Python, se debe importar el módulo Android e instanciar un objeto de la siguiente forma:

```
import android
droid = android.Android()
```

Las funciones de la API de Android a las que se puede acceder desde los lenguajes soportados por la herramienta SL4A, las podemos encontrar en la página del proyecto [13].

- **Iperf**

Iperf [29] es una herramienta que se emplea para generar tráfico TCP o UDP y medir el ancho de banda de dicho tráfico. Permite establecer varios parámetros que permiten evaluar la calidad del enlace. Iperf funciona como una aplicación cliente-servidor, midiendo el ancho de banda entre dos puntos, en un sentido de la comunicación o en los dos. Es una herramienta de código abierto y disponible para varias plataformas, como Linux, Unix y Windows.

Según el tráfico generado sea TCP o UDP, hay que tener en cuenta las siguientes consideraciones:

- *Medidas con TCP*: Se mide el ancho de banda que se alcanza entre dos extremos. En TCP el ancho de banda se ve limitado por varios factores: pérdidas, congestión, saturación de buffer y entregas fuera de orden.

- *Medidas con UDP*: UDP permite mayor transparencia. Se puede especificar el tamaño del datagrama y obtener más resultados además del ancho de banda, como el número de paquetes perdidos y el retraso.

iPerf para Android es una aplicación gratuita que se puede descargar desde Google Play, el mercado de aplicaciones de Android [30]. Esta aplicación permite medir el rendimiento de las interfaces de datos de un dispositivo Android, generando tráfico TCP o UDP. Iperf requiere dos dispositivos: uno actuando como un servidor y otro como un cliente. iPerf para Android permite a los dispositivos funcionar como cliente o servidor, por lo que se puede utilizar entre dos dispositivos Android, aunque no es lo ideal, ya que el ancho de banda quedará limitado por el ancho de banda del dispositivo más lento.

Figura 19. Aplicación iPerf para Android [30]

En este proyecto, Iperf se ha utilizado para transmitir tráfico desde el dispositivo Android y hacia el dispositivo. En los casos en los que se genera tráfico desde el teléfono, este funciona como cliente y el ordenador como servidor. En el caso de que el teléfono reciba el tráfico, este funciona como servidor y el ordenador como cliente. Además, el tráfico generado es UDP.

3.1.3 Aplicación para la monitorización de la batería

Las medidas sobre el teléfono se toman vía software, aprovechando las capacidades que nos ofrece Android. Por un lado, se utiliza la herramienta SL4A (*Scripting Layer for Android*) para crear scripts que establecen las condiciones adecuadas para obtener el consumo de cada componente. Por otro lado, se ha desarrollado una aplicación que monitoriza la batería en todo momento y registra todos los cambios del nivel y el voltaje para calcular la potencia consumida.

Para entender la aplicación es necesario explicar el concepto “servicio” en Android. Un servicio permite a una aplicación realizar una tarea en segundo plano, incluso cuando el usuario no está interactuando con la aplicación. Un servicio puede comunicarse con el sistema Android o con otras aplicaciones.

La aplicación desarrollada es muy sencilla, no necesita interfaz de usuario, simplemente cuando se ejecuta por primera vez crea un servicio. Este servicio registra los datos del estado de la batería en un fichero de texto que se guarda en la memoria SD externa del teléfono.

El código fuente de la aplicación se compone de dos clases Java: `BatteryMonitor.java` y `BatteryService.java` (código completo en el Anexo F). En la clase `BatteryMonitor.java` se define la actividad que se ejecuta cuando la aplicación se crea. Esta actividad inicia el servicio que va a ejecutarse en segundo plano y por simplicidad no define ninguna interfaz de usuario ya que no es necesaria.

La clase `BatteryService.java` define el servicio y las operaciones que realiza. Cuando el servicio se crea por primera vez registra que cuando el sistema envíe el evento ACTION_BATTERY_CHANGED, indicando que el estado de la batería ha cambiado, la aplicación sea notificada. Por otro lado se indican los pasos a seguir cuando se produce la acción ACTION_BATTERY_CHANGED. En este caso se obtienen los datos actuales del estado de la batería, voltaje, nivel y escala, y se vuelcan a un archivo de texto en la tarjeta de memoria externa SD.

Para obtener la información del estado de la batería se ha utilizado la clase BatteryManager.[5] Esta clase ofrece funciones para conocer datos de la batería, como el nivel, el voltaje, la tecnología, la temperatura, la salud, si está cargándose, etc. La aplicación desarrollada, obtiene el nivel, el voltaje, la escala y la temperatura. El nivel, registrado en la variable EXTRA_LEVEL y voltaje, en la variable EXTRA_VOLTAGE, se necesitan para el cálculo de la potencia consumida. La temperatura se registra en la variable EXTRA_TEMPERATURE y se utiliza para comprobar si en algún momento hay un recalentamiento de la batería, ya que en este caso las medidas no serían fiables. La escala se consulta a través de la variable EXTRA_SCALE e indica el porcentaje de capacidad disponible respecto a la capacidad total teórica. La escala es necesaria para saber si la batería dispone de la capacidad total indicada en su etiqueta o si debido al deterioro la capacidad total es menor.

Esta aplicación se ha decidido realizar como un servicio que se ejecuta en segundo plano ya que mientras no se produzca ningún cambio en el nivel de la batería, la aplicación no consume CPU ni ningún otro recurso, siendo el impacto en el consumo de la batería mínimo. Por otro lado, también se decidió no realizar ninguna interfaz de usuario y obtener los datos de la batería en un fichero de texto en memoria, también para que el impacto en el consumo fuera el menor posible y además, se pueda tener un fácil acceso a los datos guardados para obtener la potencia consumida.

[5] http://developer.android.com/reference/android/os/BatteryManager.html

3.2 Metodología de medida

La evaluación del consumo de energía del dispositivo se ha realizado a nivel de componente, es decir, se ha obtenido el consumo de cada componente por separado. Es necesario aclarar que en este proyecto cuando hablamos de un componente del dispositivo bajo estudio, no nos referimos al componente hardware o chip, nos referimos a una funcionalidad como puede ser la comunicación Wi-Fi o la pantalla.

Las medidas se han tomado a nivel de dispositivo, manteniendo un cierto componente activo y el resto inactivos, de forma que se puede estimar la potencia consumida por el componente que se encuentra activo.

Se considera que la potencia total consumida por el dispositivo es la suma de cada una de las potencias consumidas por los componentes que están activos. Por lo que, en el caso de que dos componentes se encuentren activos, conociendo la potencia toatal consumida por los dos componentes y la de uno ellos, podemos obtener la potencia consumida por el otro componente.

Para obligar que ciertos componentes permanezcan activos durante todo el tiempo que dure la medida, se hace uso de los *wakelocks* de Android (ver sección 2.1.3). Esto es necesario porque si un componente está activado pero se deja de usar, Android mediante su mecanismo de ahorro de energía, lo apaga hasta que sea necesario volver a utilizarlo. Un ejemplo sería la pantalla, mientras estamos interactuando con el teléfono a través de ella permanece encendida, pero una vez que la dejamos de tocar pasado un tiempo se apaga, por lo que deberíamos usar un wakelock para evitar que la pantalla se apague mientras se realiza la medida.

Los componentes que se van evaluar son: CPU, pantalla, Wi-Fi y 3G. Se ha decido evaluar el consumo de la CPU y la pantalla, ya que son componentes imprescindibles en el uso de un dispositivo móvil. Por otro lado, se ha decidido estudiar el consumo de Wi-Fi y 3G, ya que la mayoría estudios previos sobre el consumo de energía en dispositivos móviles concluyen que la mayor parte de la potencia consumida se debe a estas dos tecnologías de comunicación, además de ser las más utilizadas por los usuarios, tal y como se vio en la introducción al proyecto.

Se han establecido diferentes escenarios según el componente a medir. Para realizar las medidas del teléfono en estado suspendido, con la CPU activa y la pantalla encendida, el escenario es simplemente el script que realiza las configuraciones necesarias durante la medida y el software de monitorización de la batería que se ejecuta en segundo plano manteniéndose el componente activo. Sin embargo, para las interfaces inalámbricas es necesario plantear escenarios más complejos para realizar el estudio del consumo.

3.2.1 Dispositivo en estado suspendido

Es necesario obtener el consumo cuando el teléfono está encendido, pero no hay ninguna tarea ni ningún componente o interfaz de datos activa. Se necesita obtener la potencia consumida en este estado, para tener una medida de referencia. De esta forma podemos obtener cuánto se incrementa el consumo al activar cada uno de los componentes evaluados.

Para obtener el consumo del dispositivo en estado suspendido, nos aseguramos, por un lado, que todas las interfaces de redes de datos se encuentran apagadas y por otro lado, que la única aplicación activa es la aplicación que se ha desarrollado para la monitorización de la batería, ejecutándose en segundo plano. Por lo tanto, en este estado no habrá ningún *wakelock* activo en el dispositivo, siendo el estado de consumo más bajo. La aplicación de monitorización de la batería, cuando se produce un cambio en el nivel de batería, recibe un evento que hará que el teléfono despierte para realizar la operación de registro de los datos de la medida, pero a continuación vuelve a suspenderse al no haber activo ningún *wakelock.*

3.2.2 CPU

La CPU es uno de los componentes que con mayor frecuencia se encuentra activo, ya que se necesita para realizar cualquier tarea. Además es uno de los componentes clave para la gestión del consumo de energía en Android, ya que si no hay ningún *wakelock* activo el administrador de energía (ver sección 2.1.3) suspende la CPU para situarse en el estado de bajo consumo.

Para obtener la potencia consumida por la CPU, por un lado se ejecuta un script con la herramienta SL4A que mantiene la CPU activa y por otro lado, la aplicación que monitoriza la batería. El script se encarga de obtener el *wakelock* que asegura que la CPU se encuentre activa durante toda la medida. El *wakelock* que mantiene encendida la CPU es el PARTIAL_WAKE_LOCK. En este script también se establece el tiempo que dura la medida y una vez finalizada, el PARTIAL_WAKE_LOCK se libera. A su vez, la aplicación de monitorización de la batería va registrando todos los datos, para obtener la potencia consumida, cuando se produce un cambio en el estado de la batería.

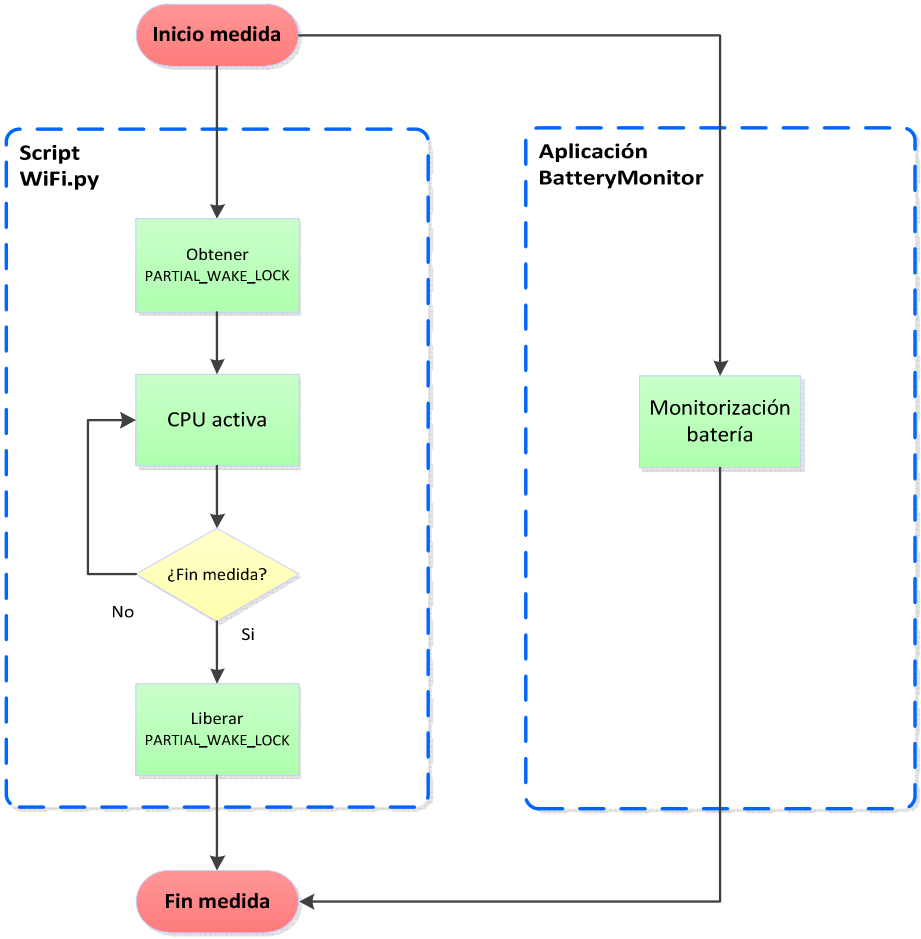

Figura 20. Diagrama flujo para medida del consumo de la CPU

3.2.3 Pantalla

La pantalla es también otro de los componentes cruciales en el consumo de un smartphone, ya que durante muchas operaciones puede estar encendida. El estudio del consumo asociado a este componente nos permitirá estimar cuanto se incrementa el consumo del dispositivo al estar encendida la pantalla.

De forma similar a la CPU, a la vez que se ejecuta la aplicación que monitoriza la batería, se ejecuta con la herramienta SL4A el script que obtiene el *wakelock* que asegura que la pantalla permanece encendida durante toda la medida. Hay tres *wakelocks* que mantienen encendida la pantalla (ver sección 2.1.3), pero en este caso se ha utilizado el SCREEN_BRIGHT_WAKE_LOCK, que establece el máximo brillo en la pantalla para estimar el máximo consumo. No utilizamos el FULL_WAKE_LOCK, ya que el terminal empleado para las medidas no dispone de teclado.

El SCREEN_BRIGHT_WAKE_LOCK también evita que la CPU se desactive, por lo que habrá que tener en cuenta, que una parte de la potencia consumida por la pantalla se debe a la tarea que se está ejecutando mientras la pantalla está encendida.

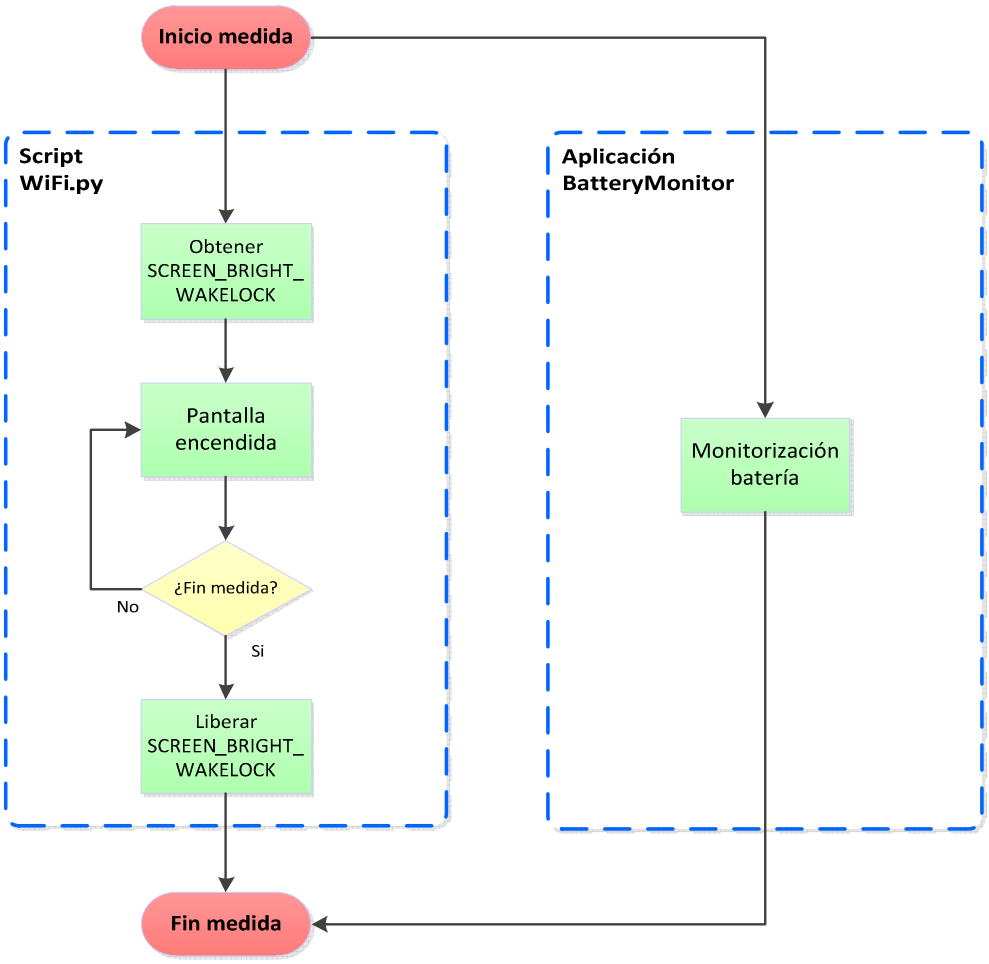

Figura 21. Diagrama de flujo para medida del consumo de la pantalla

3.2.4 Wi-Fi

La potencia consumida cuando se utiliza la tecnología de comunicación Wi-Fi, se va a estudiar según el modo de operación de la interfaz IEEE 802.11. Tal y como se vio en la sección 2.3.1.1, el protocolo PSM define cuatro modos de operación para la tarjeta inalámbrica. Teniendo en cuenta estos modos, se han definido los siguientes estados en los que se va a estudiar el consumo de la interfaz inalámbrica IEEE 802.11:

- **No asociación**: La interfaz se activa pero no se establece la asociación con un AP. En teoría, el consumo en este estado debe ser el mínimo ya que sólo se mantiene la interfaz IEEE 802.11 del dispositivo activa.

- **Escaneo**: La interfaz no está asociada a ningún AP y está constantemente realizando el escaneo de las redes disponibles. Se espera que en este estado se incremente la potencia consumida ya que además de mantener la interfaz activa se realiza una operación de búsqueda y se produce un intercambio de paquetes.

- **Asociación**: La interfaz está asociada a un AP y preparada para transmitir o recibir datos. Teóricamente, el consumo debe incrementarse respecto al estado no asociado ya que debe mantenerse la asociación con el AP.

- **Recepción**: La interfaz recibe continuamente datos. Se espera que la potencia consumida se incremente respecto a los estados anteriores debido al tráfico de datos.

- **Transmisión**: La interfaz está continuamente enviando datos. En teoría el consumo será mayor que en los estados de no asociación, escaneo y asociación, aunque habrá que estudiar si el consumo es igual, mayor o menor respecto al estado de recepción.

El diagrama de flujo genérico que se sigue para evaluar el consumo de cada uno de los estados descritos es el siguiente:

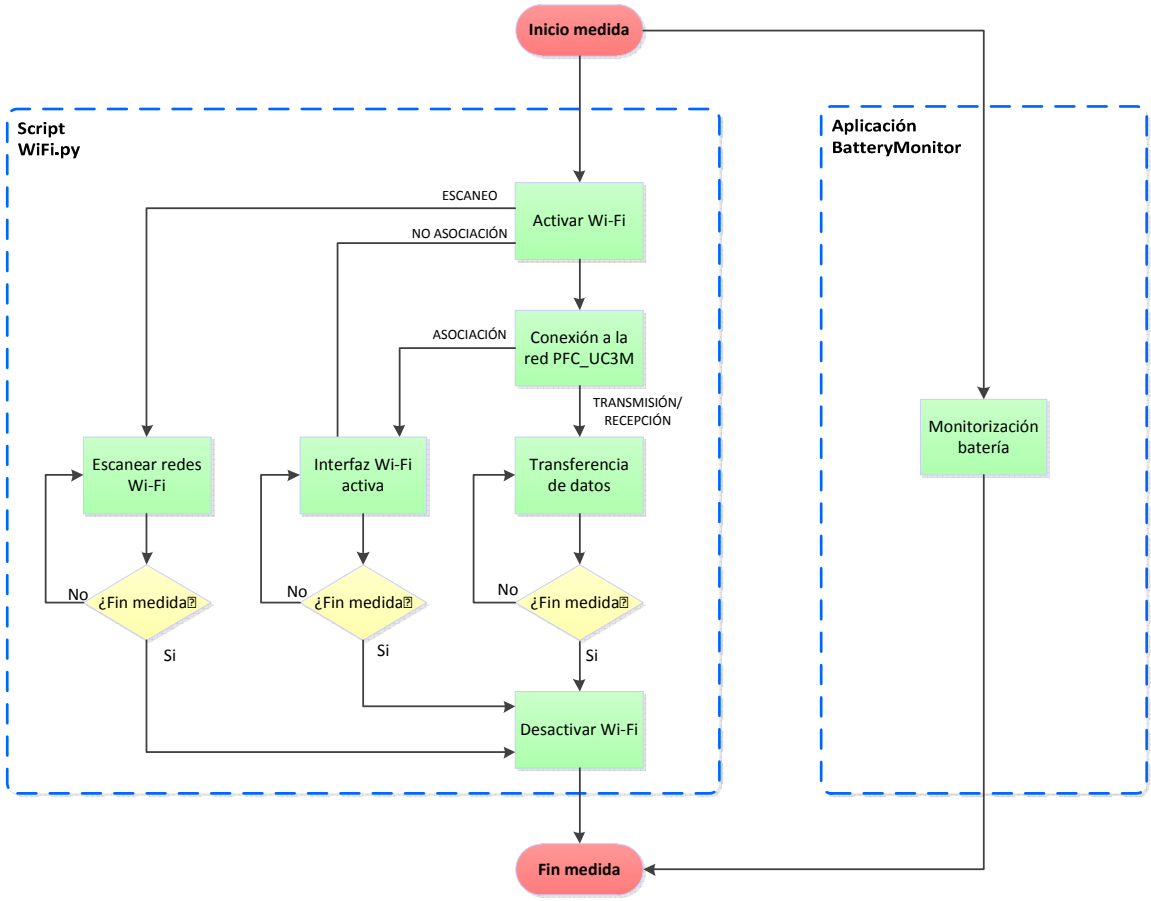

Figura 22. Diagrama de flujo para medida del consumo Wi-Fi

Por otro lado, en los estados en los que hay tráfico de datos se va a estudiar como varía el consumo según la velocidad de transferencia de los datos. En teoría, se espera que el consumo sea más alto cuanto mayor sea el ancho de banda debido al coste de manejar mayor volumen de datos.

Además, se ha definido un escenario de trabajo en el que se plantea una red Wi-Fi muy sencilla. No se ha podido utilizar una de las redes ya desplegadas en el laboratorio de trabajo, ya que se necesita poder manipular algunas variables de la comunicación para estudiar su influencia en la potencia consumida debido a las comunicaciones con Wi-Fi.

El escenario que se muestra la Figura 23, se compone del equipo de desarrollo con su interfaz inalámbrica configurada como un punto de acceso, y el dispositivo Android. La red Wi-Fi emplea un direccionamiento privado y la conexión con Internet se realiza a través de la conexión Ethernet del ordenador. El terminal Android y el punto de acceso se encuentran en la misma subred por lo que se comunican directamente. En el Anexo E, se describen los pasos que se han seguido y las configuraciones necesarias para plantear la red descrita. El enlace Wi-Fi funciona en la banda de frecuencia de 2.4 GHz, la potencia transmitida por el AP es 27dBm y la máxima velocidad teórica de transmisión es 54 Mbit/s.

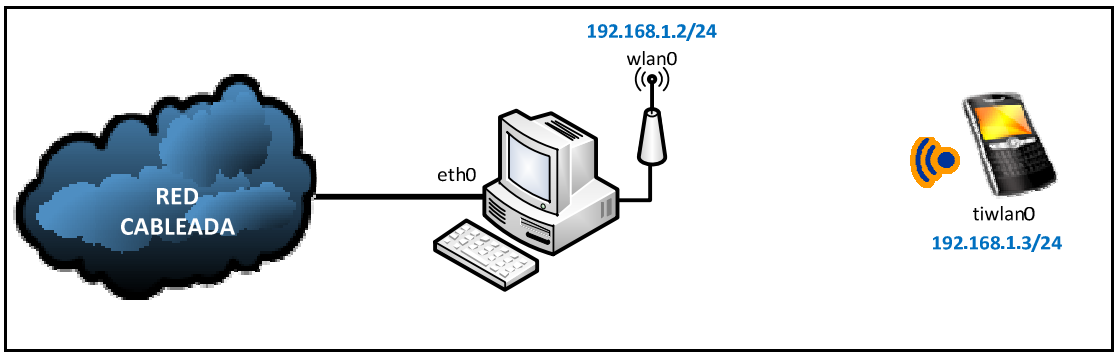

Figura 23. Escenario de trabajo con la tecnología de comunicación Wi-Fi

3.2.5 3G

Al igual que se ha definido para el caso de la tecnología Wi-Fi, para evaluar la potencia consumida por la conexión de datos 3G, se han definido distintos modos de operación en los que se espera obtener distintos niveles de consumo.

- **Inactividad**: La conexión 3G se habilita, pero permanece inactiva ya que no hay tráfico de datos. Según el protocolo RCC que se describió en el apartado 2.3.2.1, cuando no existe transferencia de datos a través de una conexión 3G, durante un tiempo igual al temporizador de inactividad, se produce una transición a un estado de bajo consumo de energía.

- **Transmisión**: El dispositivo transmite continuamente datos. Se espera que este sea un estado de alto consumo, ya que según define el protocolo RCC cuando hay tráfico de datos el consumo de energía es alto.

- **Recepción**: El dispositivo recibe continuamente datos. Al igual, que en el modo de transmisión se espera un alto consumo de potencia debido a la transferencia de datos, aunque a priori se desconoce si la potencia consumida será menor, mayor o la misma respecto a la transmisión.

A continuación se muestra el diagrama de flujo para evaluar la potencia consumida por la tecnología de comunicación 3G, según los estados anteriormente definidos:

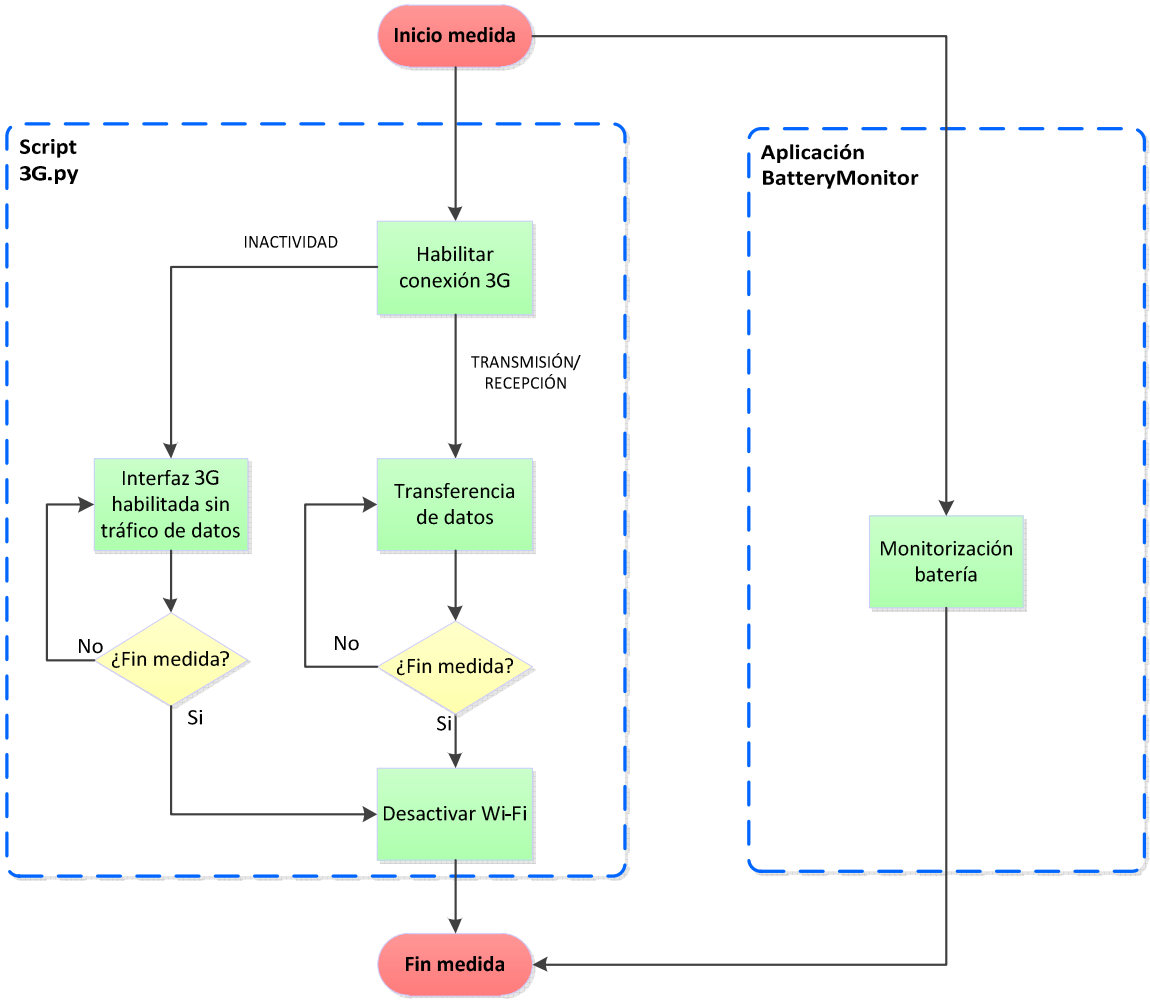

Figura 24. Diagrama de flujo para medida del consumo 3G

Para esta tecnología se ha utilizado la red desplegada por el operador de red Orange, por lo que el escenario de trabajo la propia red de este operador. Para acceder a esta red se utiliza una tarjeta SIM con tarifa plana diaria para acceso a Internet. Cuando la velocidad es máxima, en teoría se pueden llegar a alcanzar 3.6 Mbit/s, pero se mide con la aplicación IPerf que la velocidad es aproximadamente 1Mbit/s. Cuando se supera el límite de datos descargados por la tarifa, la velocidad de datos baja a los 64 kbit/s.

Por otro lado, se han realizado las medidas con el teléfono en la misma posición para asegurarnos que el nivel de señal recibida en todas las medidas es siempre el mismo. El nivel de señal recibida en el dispositivo es de aproximadamente -95 dBm, no muy alto al tener cobertura limitada en el laboratorio de trabajo.

Capítulo 4

Medidas de consumo energético

En este capítulo se muestran las medidas que se han realizado para evaluar el consumo de un smartphone HTC Legend con sistema operativo Android, según la metodología descrita en el capítulo anterior.

Los resultados del consumo del dispositivo Android se han dividido en dos partes. Por un lado, se ha estudiado como se descarga la batería según el componente estudiado. De esta forma, se puede observar cómo afecta el uso de cada uno de los componentes a la vida de la batería.

Por otro lado, se presentan medidas cuantitativas en forma de potencia consumida. Estos resultados permiten evaluar qué componentes del dispositivo consumen mayor cantidad de potencia.

En último lugar, se trata de validar los resultados obtenidos comparando las medidas obtenidas con otros estudios.

4.1 Estudio de la descarga de la batería

Se ha realizado un estudio sobre la descarga completa de la batería (desde el nivel de capacidad 100% hasta el 0%) para evaluar en qué forma afecta el consumo de cada uno de los componentes estudiados a la vida de la batería. Este estudio también se realiza con el objetivo de observar si existen diferentes comportamientos de descarga a lo largo de la vida de la batería, de forma que se tengan en cuenta al obtener las medidas de la potencia consumida. Además, viendo la velocidad a la que decrece el nivel de batería podemos saber qué componentes consumen más energía y cuales menos.

Inicialmente, la batería del teléfono se encuentra completamente cargada, siendo su nivel el 100% de la capacidad disponible. Se ejecuta la aplicación de monitorización de la batería y el script que establece las condiciones durante toda la medida según el componente a evaluar. La aplicación de monitorización de la batería permanece activa y registra todos los cambios de nivel de batería hasta que esta se descarga completamente.

Para las tecnologías de comunicación en los casos en los que hay transferencia de datos, además se utiliza la aplicación iPerf que actúa como cliente o como servidor según el sentido de la comunicación. Según la tecnología a evaluar se establecen las siguientes condiciones de comunicación para los estados en los que hay transferencia de datos:

- ***Wi-Fi***: Se estable como velocidad de transferencia de datos 24 Mbit/s, aunque con iPerf se mide que el ancho de banda de la comunicación es 11.5 Mbit/s.

- ***3G***: Inicialmente, durante los 250 MB transferidos la velocidad es la máxima permitida por el operador según la tarifa (aproximadamente 1Mbit/s medido con la aplicación iPerf).

Para ambas tecnologías, el teléfono se sitúa siempre en el mismo lugar para intentar que el nivel de señal recibido sea aproximadamente el mismo.

4.1.1 Dispositivo en estado suspendido

En primer lugar, se estudia cómo se descarga la batería del dispositivo cuando ningún componente se encuentra activo y el teléfono está en estado suspendido. Esta primera medida es necesaria con el objetivo de tener una medida de referencia para ver cuánto se incrementa el consumo cuando se activa un componente.

La figura de la descarga de la batería del teléfono en estado suspendido se presenta a continuación.

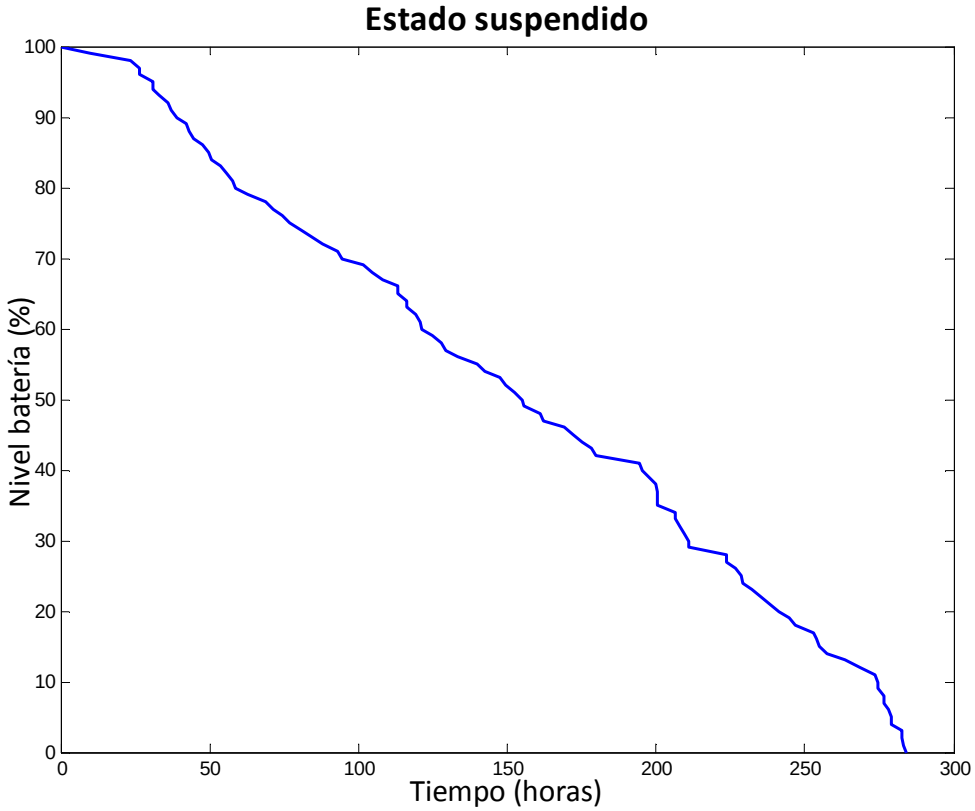

Figura 25. Descarga de la batería del dispositivo en estado suspendido

En la anterior figura podemos ver que la vida de la batería es bastante larga cuando ningún componente se encuentra activo, llegando a las 284 horas, casi 12 días. Además, podemos observar varios comportamientos diferentes en la descarga. En un principio, hasta que la capacidad de la batería no baja alrededor del 95%, la descarga es muy lenta. En oposición, el nivel de batería cae con mayor rapidez a partir del 10%. En la zona intermedia vemos que se producen pequeños escalones, pero en líneas generales se podría decir que la caída del nivel de la capacidad de la batería es aproximadamente lineal.

Si se realiza la regresión lineal por mínimos cuadrados entre el 90% y el 10% de la curva de la Figura 25, se obtiene que el coeficiente de regresión r^2 es 0.972. Este valor es bastante próximo a 1, por lo que la aproximación lineal que se puede hacer del comportamiento de la batería entre el 90% y el 10% de su nivel cuando el dispositivo está en estado suspendido es bastante buena[6].

[6] Aunque no se indique en las siguientes figuras de descarga de batería, los coeficientes de regresión r^2 que se obtienen entre el 90% y el 10% del nivel de batería, son muy similares al obtenido en la Figura 25.

4.1.2 CPU

Al encontrarse un componente activo se espera que la vida de la batería sea mucho más corta respecto al estado suspendido. En la Figura 26, podemos ver la figura de la descarga de la batería cuando la CPU está activa.

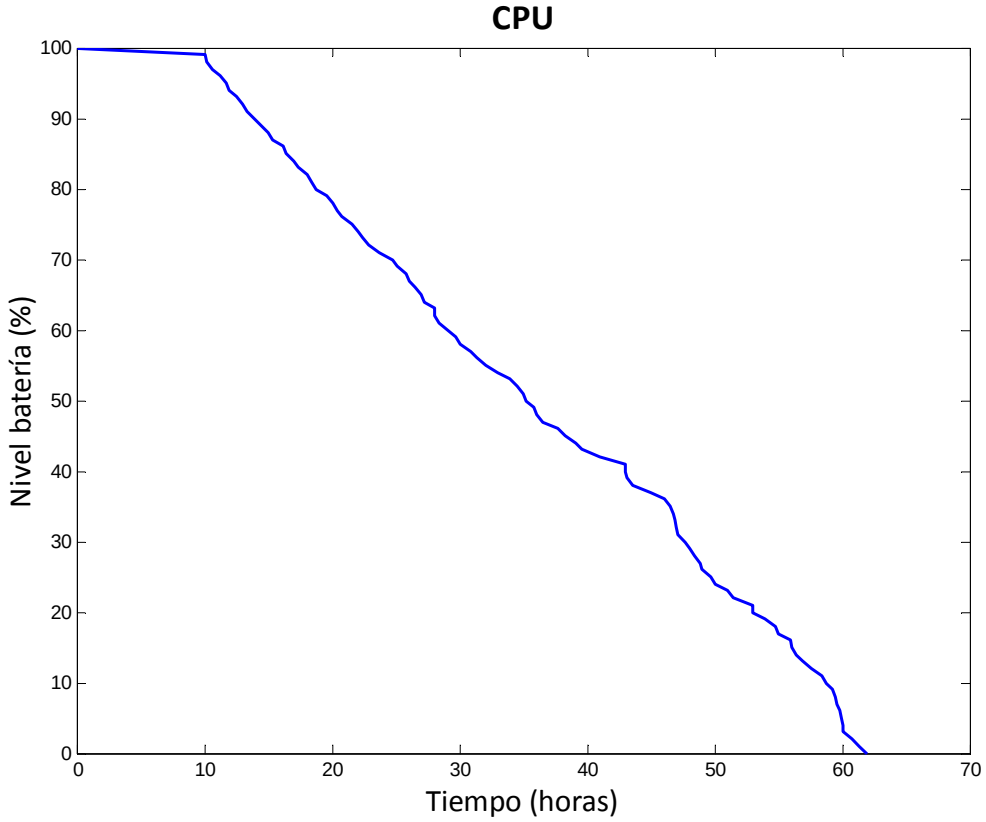

Figura 26. Descarga de la batería cuando la CPU está activa

Según la figura anterior, se puede afirmar que con el funcionamiento de la CPU, la vida de la batería se reduce notablemente respecto al estado suspendido, siendo de 62 horas. Por otro lado se observa que el nivel de capacidad de aproximadamente el 100% se mantiene durante un largo tiempo respecto al resto de niveles, y luego la descarga es mucho más rápida.

4.1.3 Pantalla

Con la pantalla encendida la figura de descarga de la batería se espera que tenga un pendiente aún mayor, ya que al estar la pantalla activa, la CPU también se encuentra funcionando. Según, el protocolo de administración de energía de Android que se describió en el apartado 2.1.3, al encenderse la pantalla el dispositivo abandona el estado de bajo consumo en el que se encuentra y la CPU se activa. Además, el wakelock que se utiliza para evitar que la pantalla se apague durante la medida, SCREEN_BRIGHT_WAKELOCK, mantiene también activa la CPU. Por lo tanto, para las medidas del consumo de la pantalla, se tendrá en cuenta que parte de este consumo se debe al funcionamiento de la CPU.

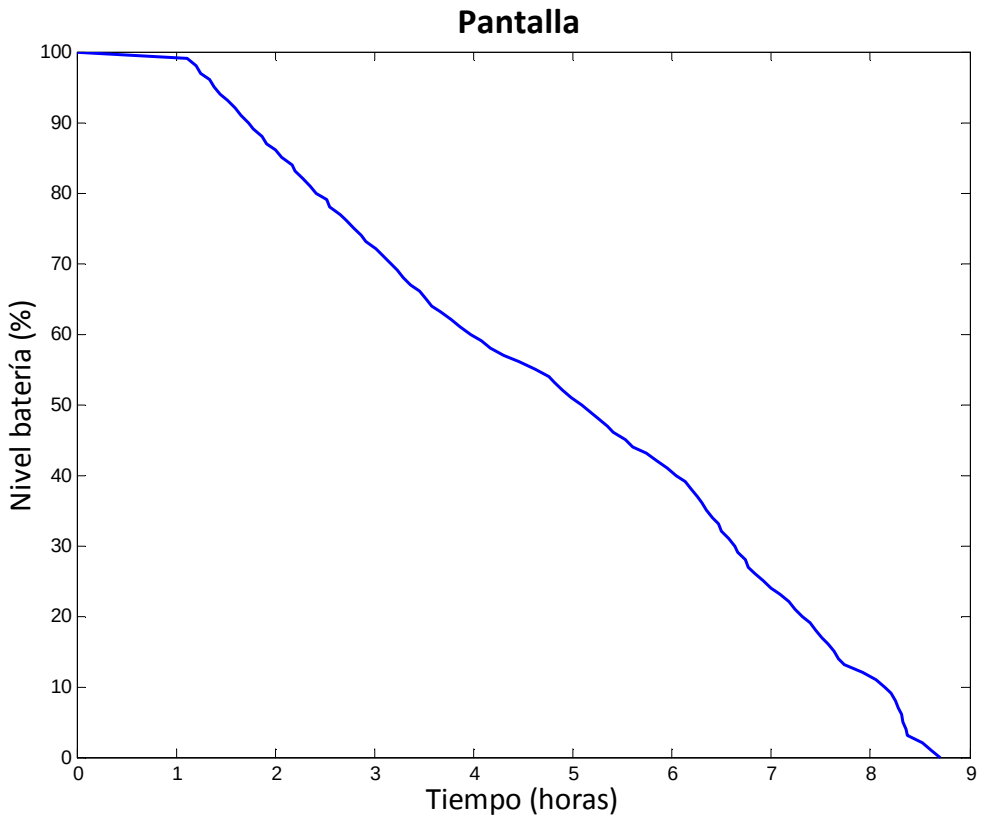

Figura 27. Descarga de la batería cuando la pantalla está encendida

En la figura anterior se ve claramente como la pantalla reduce drásticamente la vida de la batería, que no llega ni a 9 horas. Si se tiene en cuenta que parte del consumo se debe a la CPU y que la duración de la batería con CPU activa es de unas 62 horas, se puede estimar que la pantalla reduce la vida de la batería 53 horas respecto a la CPU. Además, se observa que para la pantalla se produce el mismo comportamiento que en el caso de la CPU al comienzo de la descarga, el nivel de capacidad del 100% se mantiene durante un tiempo mucho mayor que el resto de niveles, por lo que aquí se puede apreciar cierta influencia del consumo de la CPU.

4.1.4 Wi-Fi

Con el objetivo de evaluar la descarga de la batería de la tecnología de comunicación Wi-Fi, vamos a analizar cada uno de los estados de consumo que se han definido para la interfaz IEEE 802.11 en la sección 3.2.4.

En la Figura 28, se muestra la curva de descarga de batería para cada uno de los estados de consumo de la tecnología de comunicación Wi-Fi.

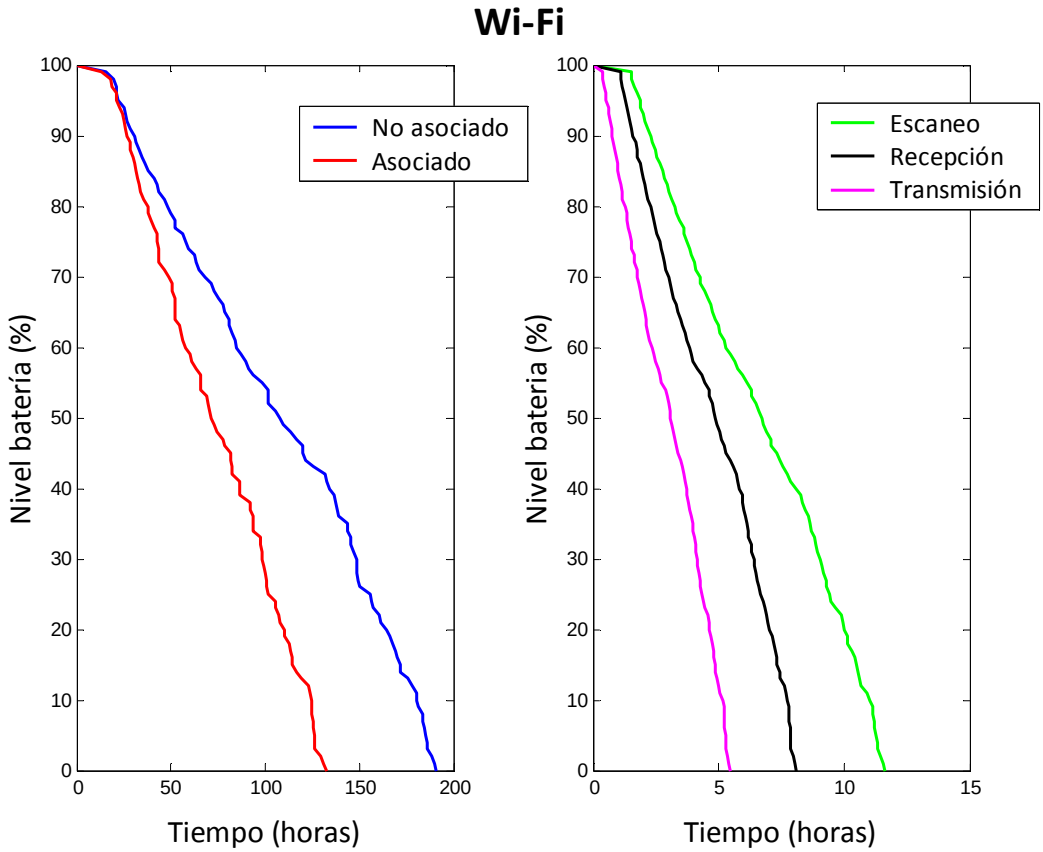

Figura 28. Descarga de la batería para los estados de consumo de la tecnología Wi-Fi

En la figura anterior podemos apreciar que, tal y como se esperaba, el estado de consumo más bajo es el de no asociación. La vida de la batería en este estado se acerca casi a las 200 horas. En el estado de asociación vemos que el consumo se incrementa, reduciendo la duración de la batería unas 60 horas. El consumo mayor en el estado de asociación se debe al coste de mantener una conexión con un punto de acceso.

Por otro lado, se observa que en el estado de escaneo, aunque no hay asociación el consumo es mayor que cuando hay asociación, ya que la batería se descarga mucho más rápido. Este hecho se atribuye al coste de transmisión en la búsqueda de redes que causa que la CPU esté activa durante esta operación.

Los estados de transmisión y recepción son los estados de más alto consumo, tal y como se ve en la Figura 28. Aunque las condiciones para la descarga de la batería han sido las mismas en transmisión que en recepción, se observa que el nivel de batería decrece con mayor velocidad en el estado de transmisión.

4.1.5 3G

De igual forma, la descarga de la batería con la tecnología de comunicación 3G se va evaluar según los modos de operación que se definieron en el apartado 3.2.5. En la Figura 29, podemos ver la descarga de la batería para los tres estados de consumo.

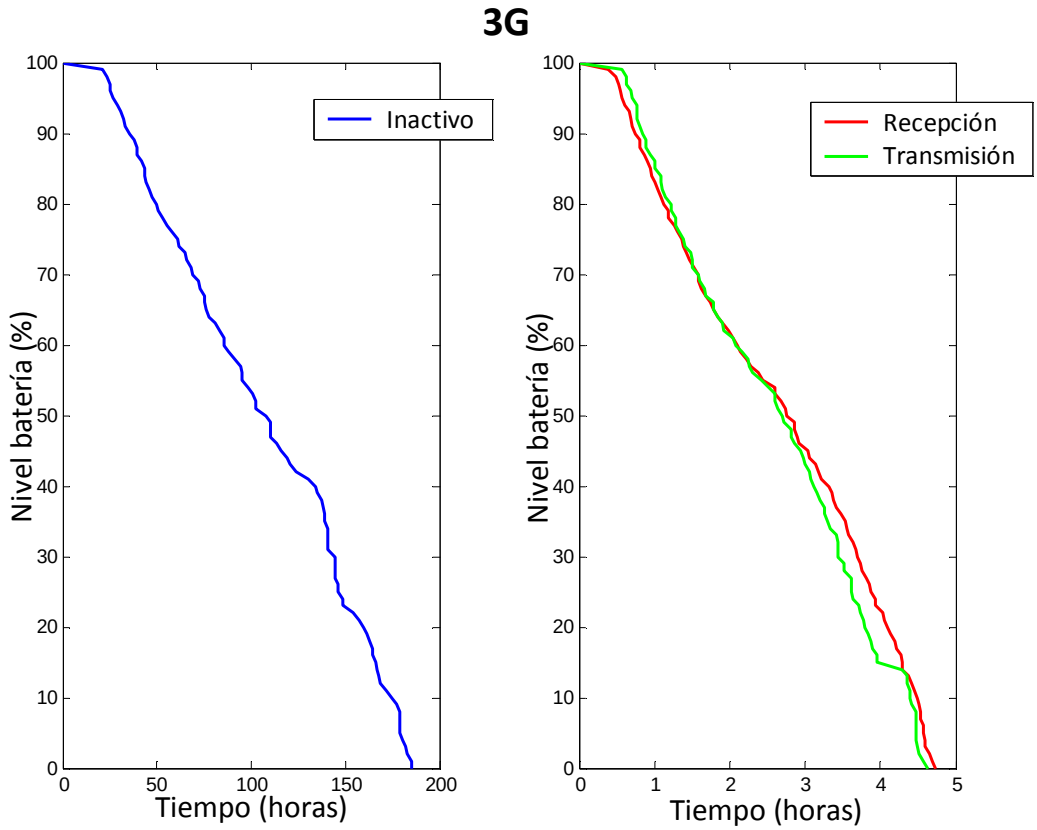

Figura 29. Descarga de la batería para los estados de consumo de la tecnología 3G

Tal y como se esperaba, el estado en el que la velocidad de descarga de la batería es más lenta es en el estado sin actividad, en el que el consumo es muy bajo. Vemos como se producen un incremento drástico en la velocidad con la que se reduce el nivel de batería en los estados de transmisión y recepción. Estableciendo las mismas condiciones en ambos estados, se obtiene que la velocidad de descarga de la batería es aproximadamente la misma en transmisión que en recepción, aunque se aprecia que es un poco más rápida en el estado de transmisión.

4.2 Medidas de potencia

A la hora de evaluar el consumo del smartphone se va a obtener la potencia consumida por el dispositivo. Como ya se ha visto, en Android se pueden realizar lecturas del voltaje y el nivel de la batería, pero no de potencia por lo que es necesario encontrar una forma de convertir los valores disponibles en potencia.

En una batería de Li-ion, el voltaje cambia durante la descarga, lo que permite estimar la potencia consumida basándonos en los cambios observados en el voltaje. A medida que cambia el voltaje va disminuyendo el nivel de capacidad de la batería, por lo que la potencia consumida se puede calcular a partir de los cambios en el voltaje, y en el nivel de capacidad de la batería. La siguiente fórmula refleja cómo obtener la potencia consumida durante un intervalo de tiempo determinado con estos datos.

$$P = \frac{C_1 \cdot V_1 - C_2 \cdot V_2}{t_1 - t_2}$$

Donde, P es la potencia media consumida en el intervalo de tiempo [t_1-t_2] en horas, C_1 y C_2 son los valores de capacidad restante de la batería expresados en mAh para los valores de voltaje V_1 y V_2 respectivamente.

Hay que tener en cuenta que la capacidad total, debe ser la real disponible ya que debido al deterioro con el paso del tiempo, la capacidad total disponible de la batería puede ser menor a la indicada por el fabricante. En Android, podemos ver cuál es la capacidad que tiene la batería en la variable `full_bat` que se encuentra en el directorio`/sys/class/power_supply/battery` del teléfono (puede variar la localización según el fabricante del dispositivo). Otra forma de conocer este valor es consultar la variable EXTRA_SCALE de la clase BatteryManager, que indica el nivel máximo disponible de la capacidad total de la batería.

En una primera fase de experimentación, se observó que si se toman cambios de voltajes pequeños que se corresponden con cambios pequeños de nivel de batería, había variaciones en las medidas cuando se repetían. Sin embargo, para cambios mayores del nivel de voltaje de la batería, la variación se reduce. Por lo que, para obtener la potencia consumida por cada componente se han considerado cambios en el nivel voltaje y por lo tanto en el nivel de capacidad de la batería, lo más grandes posibles dentro las medidas disponibles, de tal forma que los valores de potencia obtenidos para un mismo componente eran consistentes. Aun así la batería no es un dispositivo exactamente lineal y siempre no se obtiene un valor exacto de potencia, por lo que para determinar el consumo medio de un componente se obtienen varias valores de potencia y se calcula el promedio de ellos. Cabe señalar que hay medidas que han sido descartadas por variar demasiado respecto a las demás, ya que en algunos momentos se producen cambios bruscos en el voltaje de batería debido a recalentamiento excesivo de ésta.

Por otro lado, tras obtener las figuras del apartado anterior se ha observado que para todos los componentes la descarga de la batería sigue un mismo patrón. En la Figura 30, se muestra la forma general de descarga de la batería cuando se encuentra un componente activo. En primer lugar, el nivel de batería 100% se mantiene durante un tiempo considerable y hasta aproximadamente el 95% el ritmo de descarga es más lento respecto al resto de niveles de batería. Este comportamiento puede deberse al efecto del proceso de carga, de forma que cuando la batería está completamente cargada tiene capacidad mayor a la nominal. Por otro lado, se observa que a partir del 10%, cuando el nivel de batería es bajo, se descarga con mayor velocidad hasta agotarse. Entre los niveles del 90% y el 10% existe un comportamiento más regular, y según lo comprobado en la sección anterior su aproximación lineal es bastante buena. Por lo tanto, las medidas para obtener la potencia consumida se toman siempre entre el 90% y el 10% del nivel de batería que es general donde se observa un comportamiento más lineal.

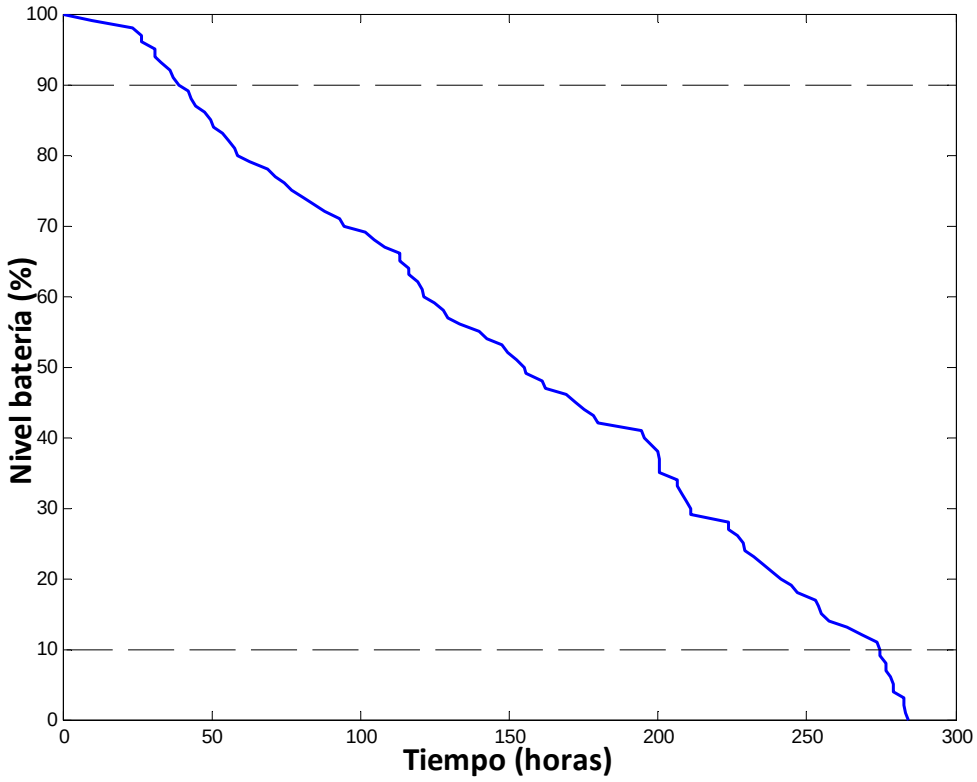

Figura 30. Figura general de descarga de la batería cuando un componente está activo

A continuación se presentan los valores de potencia consumida obtenidos según lo explicado anteriormente, por cada uno de los componentes bajo estudio del dispositivo evaluado.

4.2.1 Medidas en estado suspendido, con CPU y pantalla

En primer lugar, se indican las medidas de potencia consumida por el dispositivo en estado suspendido, cuando la CPU está activa y cuando la pantalla está encendida junto con la CPU (Tabla 4). Estas medidas son una referencia para otras medidas en las que se ven involucrados alguno de estos componentes.

Tabla 4. Medidas de potencia en estado suspendido y con CPU y pantalla activos

COMPONENTE ACTIVO	POTENCIA MEDIDA
-	18 mW
CPU	100 mW
Pantalla + CPU	660 mW

De estos primeros valores se puede concluir que el dispositivo por el simple hecho de estar encendido sin que ninguna tarea se esté ejecutando, ni ningún componente esté activo consume 18 mW. Este valor será considerable para componentes cuyo consumo sea bajo, pero despreciable para componentes con gran consumo de potencia. Para el caso de la CPU, este valor no es despreciable. Aunque se ha obtenido un valor de 100 mW, se puede aproximar que realmente la potencia consumida por la propia CPU es de 82 mW.

Como vemos en la Tabla 4, cuando la pantalla está encendida y a su vez la CPU se encuentra activa, la potencia consumida es 660 mW. Teniendo en cuanta los valores de potencia cuando no hay ningún componente activo y cuando solo la CPU se encuentra activa, se estima que la potencia consumida por la pantalla es 560 mW.

4.2.2 Medidas con Wi-Fi

Para la tecnología de comunicación Wi-Fi, se ha obtenido la potencia consumida por el dispositivo en cada uno de los estados de consumo que se han definido.

Tabla 5. Medidas de potencia en los estados de consumo Wi-Fi

ESTADO DE CONSUMO	POTENCIA MEDIDA
No asociación	29 mW
Asociación	47 mW
Escaneo	492 mW
Transmisión	1000 mW
Recepción	830 mW

En primer lugar, debemos tener en cuenta que la potencia consumida por el dispositivo cuando ningún componente está activo es 18 mW. Por lo tanto se puede aproximar que la potencia consumida por la interfaz IEEE 802.11 en el estado de no asociación, es decir, cuando la interfaz está habilitada pero no se ha asociado a ningún punto de acceso, es 11 mW. Un valor muy bajo, tal y como se estimó en un principio.

El consumo de la interfaz IEEE 802.11 es algo mayor cuando la interfaz establece una asociación con un AP. Se aproxima que el consumo en el estado de asociación es 29 mW. Este valor de potencia consumida es algo mayor que en el estado de no asociación, aunque sigue siendo un valor bastante bajo, que es lo que se esperaba obtener.

Sin embargo, cuando la interfaz IEEE 802.11 realiza la operación de escaneo el consumo se incrementa significativamente. Hay que tener en cuenta que en el valor de potencia obtenido se incluye, aparte de la referencia de 18 mW la potencia consumida por la CPU que se encuentra activa durante la operación de búsqueda de redes Wi-Fi. Por lo tanto, se estima que en el estado de escaneo la potencia consumida por es 392 mW.

El consumo de potencia se dispara en los estados de transmisión y recepción. Ya que los valores de potencia obtenidos son altos, cuando hay transferencia de datos, la potencia consumida por la interfaz por el hecho de estar activa (11 mW) se puede prácticamente despreciar. También, es importante destacar que potencia consumida en estos estados es la suma de potencia consumida por la interfaz para la transmisión o recepción, y la potencia consumida por la CPU, debido al coste computacional del procesado de los datos. Este coste computacional incluye el coste de copiar los datos entre el espacio de usuario, el kernel y la interfaz Wi-Fi. Por lo tanto, se estima la potencia consumida por la interfaz IEEE 802.11 en transmisión 900 mW y en recepción 730 mW.

Por otro lado, se va evaluar el impacto del volumen de datos recibidos en la potencia consumida. En la Tabla 6, se indican las medidas correspondientes a diferentes tasas de descarga de datos.

Tabla 6. Medidas de potencia para diferentes velocidades de descarga de datos

ESQUEMA DE TRANSMISIÓN	VELOCIDAD REAL DE TRANSFERENCIA[1]	POTENCIA MEDIDA	EFICIENCIA
1 Mbit/s	896 Kbit/s	550 mW	1.6 Mbits/J
2 Mbit/s	1.63 Mbit/s	590 mW	2.7 Mbits/J
5.5 Mbit/s	3.77 Mbit/s	795 mW	4.74 Mbits/J
11 Mbit/s	5.63 Mbit/s	815 mW	6.9 Mbits/J
24 Mbit/s	11.5 Mbit/s	830 mW	13.85 Mbits/J

[1]Las velocidades reales de transferencia se han medido con Iperf

Se puede observar que existe una tendencia a incrementarse la potencia consumida a medida que se incrementa la velocidad de descarga de datos, aunque para velocidades más altas las diferencias no son muy significativas. A mayor cantidad de volumen de datos el coste computacional es mayor lo cual implica mayor nivel de potencia consumida. Sin embargo, la eficiencia, en bits/J, es mejor para las velocidades de

transferencia más altas, por lo que para transmitir un determinado tamaño de datos el consumo sería mayor para velocidades de transferencia menores.

4.2.3 Medidas con 3G

Las medidas de potencia consumida que se han obtenido en los estados de consumo definidos para la tecnología de comunicación 3G, se presentan en la siguiente tabla.

Tabla 7. Medidas de potencia en los estados de consumo 3G

ESTADO DE CONSUMO	POTENCIA MEDIDA	EFICIENCIA[1]
Inactivo	33 mW	-
Transmisión	1300 mW	769.23 Kbits/J
Recepción	1200 mW	833.33 Kbits/J

[1] La velocidad de transferencia medida con Iperf en 3G es 1 Mbit/s

En el estado de inactividad la potencia consumida es muy baja. Teniendo en cuenta el valor de potencia de referencia, la potencia consumida por la interfaz radio 3G cuando se habilita pero no existe conexión es 15 mW.

En la Tabla 7, se observa que el consumo en los estados de transmisión y recepción se incrementa drásticamente. Vemos que en este caso, la potencia consumida por el dispositivo en estos estados en los que hay transferencia de datos es casi la misma, siendo un poco mayor en transmisión. También se aprecia que aunque la eficiencia de la transferencia de datos es parecida en transmisión y recepción, el algo mejor en transmisión.

Tal y como señalábamos en Wi-Fi, la potencia consumida en los estados de transmisión y recepción se divide entre la potencia consumida por la interfaz radio y la potencia consumida por la CPU. Por lo tanto, se estima que en transmisión la potencia consumida es 1200 mW y en recepción la potencia consumida por la interfaz radio 3G es 1100 mW.

4.3 Validación de resultados

En esta sección por un lado, se va a realizar una comparativa entre los componentes evaluados, para definir cuáles son los de mayor o menor consumo. Por otro lado, se va intentar validar los resultados obtenidos, realizando una comparación con los resultados de otros estudios similares a este proyecto. La mayoría de los estudios se centran en el consumo de las tecnologías de comunicación por lo que para ellas se van a tener más datos para validar.

En la Figura 31, se puede apreciar los niveles de potencia consumida por los diferentes componentes evaluados. En la parte de arriba de la figura se muestran los componentes y estados de consumo que tienen niveles de potencia más reducidos y en la parte inferior de la figura se indican los de mayor consumo.

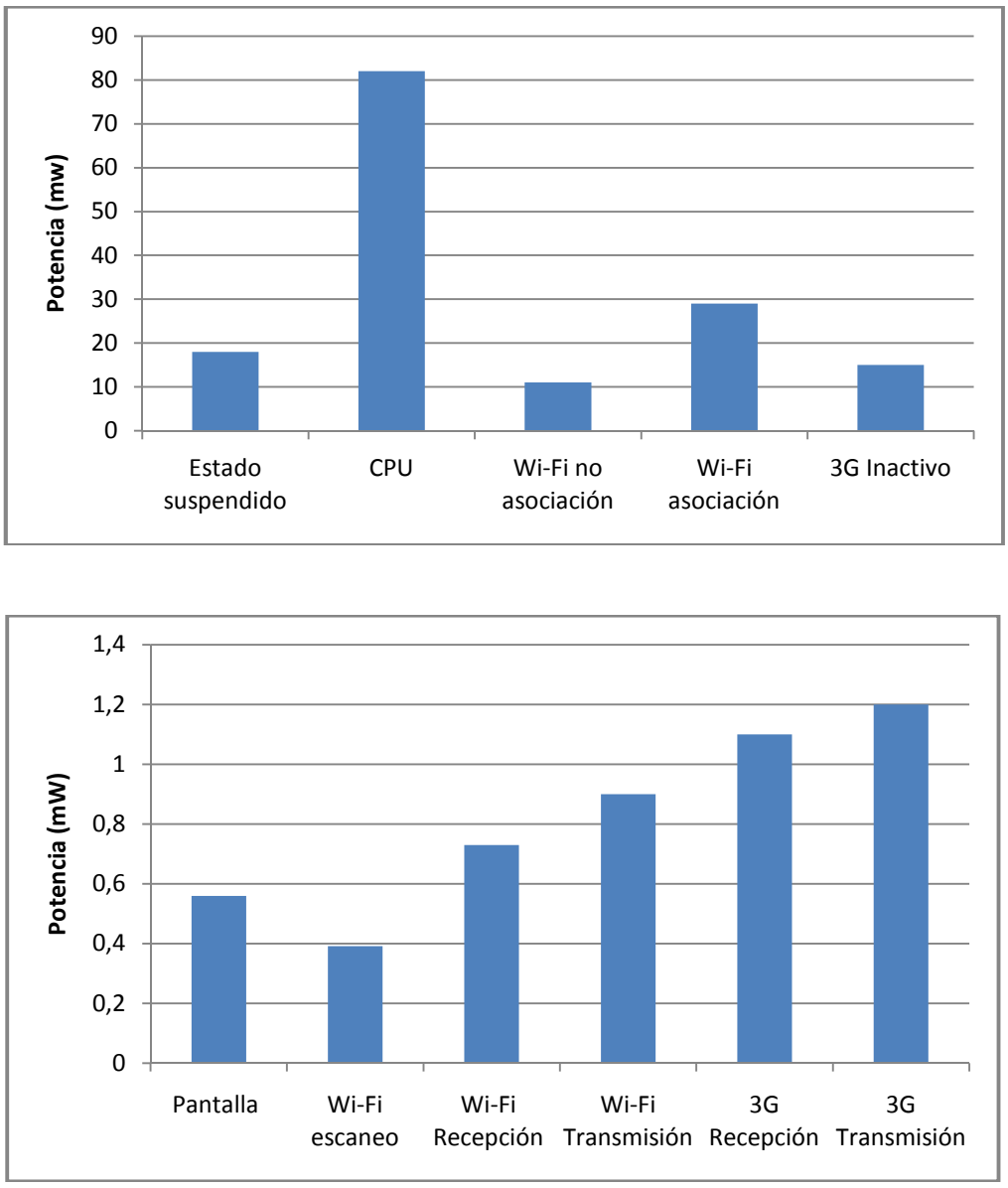

Figura 31. Niveles de potencia consumida por cada componente

Por otro lado, en la Tabla 8 se indican los valores obtenidos de potencia consumida según las aproximaciones que se han realizado y justificado, para cada uno de los componentes en cada uno de sus estados de consumo. A partir de esta tabla y la Figura 31, se puede concluir que 3G es el componente de todos los estudiados que consume mayor potencia. Wi-Fi, la otra tecnología de comunicación estudiada, consume niveles de

potencia muy altos y aunque son menores que con 3G, también son muy elevados. Se observa que para ambas tecnologías, el consumo en el estado de transmisión es mayor que en recepción. También podemos ver que en los estados en los que las interfaces 3G y Wi-Fi están habilitadas sin transferir datos, la potencia consumida no es muy alta, pero contribuye a que el consumo sea mayor en comparación con el hecho de que no estuvieran activos.

Por otro lado, el valor de la potencia consumida por la pantalla es bastante alto, lo cual incrementa significativamente el consumo.

El componente de menor consumo de los estudiados es la CPU. Aunque el valor de potencia consumida no es muy alto, incrementa el consumo de forma significativa respecto al estado suspendido.

Tabla 8. Potencia consumida por cada uno de los componentes estudiados

COMPONENTE ACTIVO		POTENCIA CONSUMIDA	POTENCIA ESTUDIOS ANTERIORES [Referencia]
-		18 mW	25 mW [6]
CPU		82 mW	100 mW [6]
Pantalla		560 mW	-
Wi-Fi	No asociación	11 mW	-
	Asociación	29 mW	20 mW [5]
	Escaneo	392 mW	-
	Transmisión	900 mW	1000 mW [5] / 730 mW [31]
	Recepción	730 mW	730 mW [31]
3G	Inactivo	15 mW	10 mW [5]
	Transmisión	1200 mW	1000 mW [31]
	Recepción	1100 mW	1000 mW [31]

En la Tabla 8 también se incluyen los valores de potencia que se han obtenido en estudios previos a este. Ya que cada uno de los estudios es diferente y se centra en ciertos componentes no se han podido obtener valores para todos los componentes o estados de consumo que se han evaluado en este proyecto. Si se comparan los valores no hay diferencias muy significativas, aunque las variaciones se pueden deber a que las condiciones y escenarios en las que se obtienen los valores de potencia no son las mismas.

En el estado suspendido, se ha definido que la potencia consumida en este estado es 18 mW. En el estudio [6], se ha encontrado que para los smartphones Android Nexus One y HTC Dream G1, la potencia consumida es en torno a los 25 mW, un resultado no muy diferente al nuestro. En el mismo estudio se obtiene, que cuando hay actividad en el dispositivo la potencia consumida por la CPU es cercana a los 100 mW, un valor cercano al se ha obtenido en este proyecto.

Para la tecnología de comunicación Wi-Fi, podemos dar evidencia de varios hechos. En primer lugar, en el estudio [2], sobre el consumo de la interfaz IEEE 802.11 realizado en ordenadores portátiles, se obtiene que en transmisión el consumo es mayor que en recepción. A este mismo resultado, se llega en este proyecto. De la mima forma, en el anterior estudio y en este proyecto, se concluye que cuanto mayor es la velocidad de transferencia de datos, el nivel de batería decrece con mayor velocidad.

Por otro lado, en el estudio realizado [5], donde se diseña un modelo para obtener la potencia consumida por un dispositivo Android, se obtiene la potencia consumida con Wi-Fi en el estado de transmisión es 1000 mW y cuando no hay tráfico 20 mW. En otro estudio [31], en el que se mide directamente la potencia consumida sobre el teléfono, se obtiene que tanto para transmisión y recepción el consumo es 730 mW. Mientras que en nuestro estudio hemos obtenido resultados similares: en el estado de transmisión el consumo es 900 mW, en recepción 730 mW y en el de asociación 29 mW.

En cuanto a 3G, en el estudio ya mencionado [5], se obtiene que cuando no hay actividad de red el consumo es 10 mW, muy similar a los 15mW que obtiene nuestro estudio. En el estudio [31], se obtiene que 3G consume en torno a 1000 mW cuando la interfaz está activa, un valor algo menor que el que hemos obtenido pero en el mismo rango.

Si comparamos las tecnologías 3G y Wi-Fi, en el estudio [3] en el que se compara el consumo de distintas tecnologías de comunicación, se obtiene que 3G consume significativamente más energía que Wi-Fi, al igual que se ha obtenido en este proyecto.

Viendo que los resultados de este proyecto son bastante similares y están en el mismo rango que los obtenidos en otros estudios, se concluye que los resultados experimentales logrados en este proyecto son válidos.

Capítulo 5

Conclusiones y trabajo futuro

Finalmente, este capítulo incluye las conclusiones obtenidas tras la realización de este proyecto. Por otro lado, se incluyen líneas de trabajo futuras que se pueden desarrollar como continuación de este proyecto.

5.1 Conclusiones

La energía en los dispositivos móviles se ha convertido en un recurso de vital importancia. Ya que estos dispositivos están integrados por gran variedad de componentes y tecnologías de comunicación, es importante entender y caracterizar su consumo relativo de energía. Con ese objetivo, este proyecto ha llevado a cabo un estudio del consumo de energía de diferentes componentes en un smartphone con sistema operativo Android.

La principal contribución de este estudio ha sido el desarrollo de una metodología que permite evaluar el consumo de distintos componentes por separado en un dispositivo Android. Las principales características de esta metodología se presentan a continuación:

- No es necesaria la interacción del usuario con el dispositivo durante las medidas.
- No se requiere ningún equipo de medida externo, ni hay que modificar el hardware del teléfono.
- Permite mantener activos diferentes componentes por separado vía software.
- Monitorización de la batería con el menor impacto posible.
- Permite el estudio de la descarga de la batería para cada uno de los componentes.
- Permite obtener la potencia consumida de diferentes componentes por separado.
- Se puede portar a otros terminales de distintos fabricantes al del dispositivo utilizado, siempre que su sistema operativo sea Android.
- Existe la limitación de que el consumo pueda variar respecto a otros dispositivos con diferentes componentes hardware que consuman niveles de energía distintos a los consumidos por los componentes del dispositivo utilizado en este estudio.

Con la metodología desarrollada, por un lado, se ha llevado a cabo el estudio de la descarga de batería de los distintos componentes evaluados, que ha permitido definir en un primer lugar, qué componentes consumen más según la velocidad a la que descargan la batería.

Por otro lado, la metodología desarrollada permite obtener valores de potencia consumida por los distintos componentes y en diferentes estados de consumo. Estos valores dan una idea más concreta de la energía consumida por cada uno de los componentes evaluados, además de ser una medida válida que permite su comparación con otras obtenidas en diferentes estudios.

Con los valores de potencia consumida y el estudio de descarga de batería, se tiene evidencia de que las tecnologías de comunicación, Wi-Fi y 3G, son los componentes que mayor potencia consumen. Otro componente que consume un alto nivel de potencia es la pantalla. El componente de todos los estudiados que tiene un consumo más bajo, es la CPU, aunque supone un consumo significativo en comparación con el estado suspendido.

En cuanto a los diferentes estados de consumo de Wi-Fi y 3G, en el estado de inactividad, en el que la interfaz está habilitada pero no se puede transferir datos, se evidencia la efectividad de los algoritmos de administración de energía, PSM y RCC respectivamente, al ser la potencia consumida en estos estados muy baja.

En Wi-Fi, gracias a la red que se ha desplegado en el laboratorio, se ha demostrado que el consumo se incrementa cuanto mayor es la velocidad de transferencia de datos, aunque la eficiencia es mayor para velocidades más altas. Además, los resultados obtenidos indican que el consumo es mayor en el dispositivo en transmisión que en recepción.

Para 3G, se obtiene que el consumo cuando hay intercambio de datos es superior a Wi-Fi. Se observa también, que en transmisión la potencia consumida es mayor que en recepción, aunque la diferencia es menos significativa respecto a Wi-Fi. Sin embargo, el coste de mantener la interfaz habilitada (estado de inactividad en 3G y no asociación en Wi-Fi) es aproximadamente el mismo para ambas tecnologías de comunicación.

Por último, se ha realizado una validación de los resultados obtenidos comparando con otros resultados obtenidos en estudios previos a este proyecto. Se ha concluido que los valores de potencia obtenidos para los distintos componentes se asemejan a los obtenidos en otros trabajos similares y por lo tanto, tienen validez.

Con este estudio, se ha caracterizado el consumo de distintos componentes de un smartphone Android. Además, se ha dado evidencia de los altos niveles de potencia consumida por los diferentes componentes evaluados respecto a no encontrarse activos. Componentes como la CPU o la pantalla, imprescindibles en la utilización de un dispositivo móvil, reducen drásticamente la vida de la batería.

En cuanto a las tecnologías de comunicación, se concluye que la utilización de Wi-Fi en un smartphone es más eficiente en cuanto a energía que 3G. Además, las velocidades de datos que se pueden alcanzar con Wi-Fi son bastantes más elevadas que con 3G. Sin embargo, Wi-Fi tiene la limitación que es una tecnología de disponibilidad muy reducida y que permite poca movilidad. Por lo tanto, la decisión de usar Wi-Fi o 3G para transferencias de datos, es un compromiso entre la disponibilidad de la red y el ahorro de energía.

5.2 Trabajo futuro

La primera línea de trabajo futuro que se propone, sería completar este estudio con la evaluación de otros componentes que no han sido evaluados en este proyecto, como Bluetooth, GPS, audio o video. Además, sería interesante caracterizar aparte del consumo de distintos componentes, el consumo de operaciones que realizan los dispositivos móviles habitualmente, en las que se ven implicados el uso de varios componentes a la vez, como la navegación a través de una página web o la visualización on-line de un vídeo.

Aunque, la metodología presentada se ha intentado desarrollar de la forma más general posible y se puede aplicar para cualquier dispositivo Android, se podría intentar desarrollar una metodología similar para obtener el consumo de energía en un dispositivo móvil con otro sistema operativo distinto a Android.

Por último, tras la realización de este proyecto, ha quedado clara la importancia de la energía en los dispositivos móviles, y la gran necesidad de una buena administración de energía en estos dispositivos. Por lo tanto, se plantea como otra línea de trabajo futuro, la

búsqueda de mecanismos o algoritmos que permitan realizar una eficaz administración de energía de forma que el consumo de los distintos componentes de un dispositivo móvil se vea reducido significativamente. Por ejemplo, se propone el desarrollo de un mecanismo para la elección de la tecnología de comunicación más eficiente en cuanto a consumo de energía en cada momento basándose en el nivel de señal. Otra posibilidad sería, cuando se tenga acceso a una red Wi-Fi, desactivar 3G y habilitar Wi-Fi.

Capítulo 6

Planificación y presupuesto

Este capítulo está dedicado, por un lado a la planificación y las fases de desarrollo para la realización de este proyecto, y por otro lado, al presupuesto, desglosado en los cotes de personal, del material y costes totales.

6.1 Planificación

A continuación, se detalla la planificación seguida para la realización de este proyecto. En concreto, el proyecto se ha desarrollado en cinco fases principales:

- **Fase 0: Documentación y estudio del estado del arte**

  Esta primera fase se centra en la búsqueda de información y el estudio de distintos aspectos necesarios para el desarrollo del proyecto. Se divide en las siguientes subfases:

  - *Subfase 0.1*: Estudio del consumo de energía en dispositivos móviles. Se busca como evaluar y medir la potencia consumida en un smartphone: metodologías, materiales, herramientas software, etc.

- *Subfase 0.2*: Estudio de la plataforma Android, centrado en el consumo y la administración de energía. Además, se buscan formas para obtener la potencia consumida en un smartphone Android.

- Subfase 0.3: Estudio sobre las formas de acceso a un dispositivo Android desde un PC Linux y las operaciones que se pueden realizar sobre él.

- *Subfase 0.4:* Estudio del desarrollo y ejecución de scripts en un dispositivo Android. Principalmente se estudia la herramienta SL4A. Además, se estudia la programación de scripts con Python.

- **Fase 1: Experimentación y obtención de primeros resultados**

  En esta fase se escriben los primeros scripts que pretenden recoger los datos necesarios para obtener la potencia consumida por distintos componentes y se analizan los primeros resultados obtenidos. Se divide en las siguiente subfases:

  - Subfase 1.1: Desarrollo de scripts para obtener consumo distintos componentes.

  - Subfase 1.2: Primeras pruebas, evaluación y análisis de los primeros resultados.

- **Fase 2: Configuración y evaluación de las tecnologías de comunicación**

  Antes de comenzar a evaluar el consumo de las tecnologías de comunicación se necesita configurar y analizar distintos aspectos, que se describen en las siguientes subfases:

  - Subfase 2.1: Configuración del escenario Wi-Fi: Estudio sobre como activar Wi-Fi y conectarse a un AP mediante línea de comandos, en lugar de con administrador de conexiones inalámbricas. Durante esta tarea se necesita rootear el HTC Legend.

  - Subfase 2.2: Estudio sobre cómo generar tráfico desde y hacia un smartphone Android.

  - Subfase 2.3: Análisis de la potencia consumida por Wi-Fi

  - Subfase 2.4: Análisis de la potencia consumida por Wi-Fi variando la tasa de transferencia de datos.

  - Subfase 2.5: Configuración del escenario y experimentación con 3G.

  - Subfase 2.6: Análisis de la potencia consumida por 3G.

- **Fase 3: Mejoras**

Viendo que los primeros resultados no son lo suficiente precisos, se realizan diferentes mejoras, que se detallan en las siguientes subfases:

- Subfase 3.1: Desarrollo aplicación de monitorización de la batería.

- Subfase 3.2: Optimización de los scripts desarrollados para cada uno de los componentes evaluados.

- Subfase 3.3: Realización de medidas para cada uno de los componentes y validación de las mismas según estudios similares.

- **Fase 4: Resultados finales**

Tras las mejoras realizadas, se procede a obtener las medidas finales:

- Subfase 4.1: Estudio de la descarga de la batería para todos los componentes evaluados

- Subfase 4.2: Obtención de medidas de potencia consumida para todos los componentes evaluados.

- **Fase 5: Memoria del proyecto**

Una vez obtenidos resultados finales se procede a documentar el desarrollo de todo el proyecto.

- Subfase 5.1: Redacción de la memoria

- Subfase 5.2: Corrección y revisión de la memoria.

Tabla 9. Resumen de la duración de las fases del proyecto

FASE	SUBFASES	DURACIÓN (semanas)
Fase 0: Documentación y estudio del estado del arte	0.1	2
	0.2	2
	0.3	1
	0.4	2
Fase 1: Experimentación y obtención de primeros resultados	1.1	2
	1.2	2
Fase 2: Configuración y evaluación de las tecnologías de comunicación	2.1	4
	2.2	1
	2.3	1
	2.4	2
	2.5	1
	2.6	1
Fase 3: Mejoras	3.1	1
	3.2	1
	3.3	2
Fase 4: Resultados finales	4.1	8
	4.2	3
Fase 5: Memoria del proyecto	5.1	7
	5.2	2
DURACIÓN TOTAL		**45 semanas**

La duración total de este proyecto ha sido de 45 semanas. Considerando que durante cada semana se han trabajado 5 días, y que cada día, se han dedicado 5 horas al proyecto, el número de horas necesarias para el desarrollo del proyecto han sido 1125 horas.

6.2 Presupuesto

A continuación se van a detallar cada uno de los costes del proyecto para obtener el presupuesto total. Se incluye la hoja de presupuesto del proyecto con los costes del personal, los costes materiales y otros costes directos.

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1.-**Autor**: Elena López Orgaz

2.- **Departamento:** Ingeniería Telemática

3.- **Descripción del Proyecto**:

- Titulo **Estudio del consumo de energía en un dispositivo Android**
- Duración (meses) **11.25 meses**

Tasa de costes Indirectos: **20%**

4.- Presupuesto total del Proyecto (valores en Euros):

27.962,51 Euros

5.-Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar - solo a título informativo)	Categoría	Dedicación (hombres mes) [a)]	Coste hombre mes	Coste (Euro)	Firma de conformidad
Elena López Orgaz		Ingeniero Junior	8,57	2.694,39	23.090,92	
		Hombres mes 8,57		**Total**	23.090,92	

[a)] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)

Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d)]
Ordenador sobremesa	700,00	100	10	60	116,67
HTC Legend	387,00	100	10	60	64,50
				Total	181,17

[d)] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} x C x D$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
	Total	0,00

OTROS COSTES DIRECTOS DEL PROYECTO[e)]

Descripción	Empresa	Costes imputable
Saldo para tarjeta SIM de conexión a Internet	Orange	30,00
	Total	30,00

[e)] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas, otros,…

6.-**Resumen de costes**

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	23.091
Amortización	181
Subcontratación de tareas	0
Costes de funcionamiento	30
Costes Indirectos	4.660
Total	**27.963**

Finalmente, se concluye que el presupuesto total de este proyecto asciende a la cantidad de **27.963 €.**

Leganés, a 8 de Octubre 2012

El ingeniero proyectista

Fdo. Elena López Orgaz

Glosario

3G	*Third Generation*
ADB	*Android Debug Bridge*
ADT	*Android Development Tools*
AP	*Access Point*
API	*Application Program Interface*
ARM	*Advance RISC Machine*
ASE	*Android Scripting Environment*
BSA	*Basic Service Area*
BSS	*Basic Service Set*
CDMA	*Code Division Multiple Access*
CN	*Core Network*
CPU	*Central Process Unit*
CSMA/CA	*Carrier Sense Multiple Access/Collision Avoidance*
DCH	*Dedicated Channel*
DS	*Distribution System*
DSSS	*Direct Sequence Spread Spectrum*
EDGE	*Enhanced Data for GSM Evolution*
FACH	*Forward Access Channel*
FDD	*Frequency Division Duplex*
FHSS	*Frequency Hopping Spread Spectrum*
GPS	*Global Position System*

GSM	*Global System for Mobile Communications*
HSPA+	*Evolved High Speed Packet Access*
IDE	*Integrated Development Environment*
iOS	*iPhone Operating System*
ITU	*International Telecommunication Union*
JDK	*Java Development Kit*
JSON	*JavaScript Object Notation*
LLC	*Logical Link Control*
MAC	*Media Access Control*
MIMO	*Multiple-Input Multiple Output*
MMS	*Multimedia Message Service*
NDK	*Native Development Kit*
OpenGL	*Open Graphics Library*
OSA	*Open System Authentication*
PSM	*Power Saving Mode*
QAM	*Quadrature Amplitude Modulation*
RCC	*Radio Resource Control*
RNC	*Radio Network Core*
RPC	*Remote Procedure Call*
SDK	*Software Development Kit*
SIM	*Subscriber Identity Module*
SKA	*Shared Key Authentication*
SL4A	*Scripting Layer for Android*
SMS	*Short Message Service*
SSL	*Secure Socket Layer*
STA	*Station*
TCP	*Transmission Control Protocol*
TDD	*Time Division Duplex*
TDMA	*Time Division Multiple Access*
TD-SCDMA	*Time Division Synchronous Code Division Multiple Access*
UDP	*User Datagram Protocol*
UE	*User Equipment*
UMTS	*Universal Mobile Telecommunication System*
UTRA	*UMTS Terrestrial Radio Access*
UTRAN	*UMTS Terrestrial Radio Access Network*

WCDMA	*Wideband Code Division Multiple Access*
Wi-Fi	*Wireless Fidelity*
WLAN	*Wireless Local Area Network*
WNIC	*Wireless Network Interface Card*

Referencias

[1] Scott, M., Renesse, R., *Consumer smartphone usage: data traffic and network usage patterns*, Analysys Mason, Mayo 2012.

[2] Lochin, E., Fladenmuller, A., Moulin, J., Fdida, S., *Energy consumption models for ad-hoc mobile terminals*, Université Pierre et Marie Curie, 2003.

[3] Balasubramanian, N., Balasubramanian, A., Venkatramani, A., *Energy Consumption in Mobile Phones: A measurement Study and Implications for Network Applications*, University of Massachusetts Amherst, 2009.

[4] Kreiman, E., *Using Learnig to Predict and Optimise Power Consumption in Mobile Devices*, Imperial Collegue of Science, 2010.

[5] Zhang, L., Tiwana, B., Qian, Z., Zhaoguang, W., Dick, R.P., Mao, Z.M., Yang, L., *Accurate Online Power Estimation and Automatic Battery Behavior Based Power Model Generation for Smartphones*, University of Michigan and Google Inc., 2010

[6] Carroll, A., Heiser, G., *An Analysis of Power Consumption in a Smartphone*, University of New South Wales, 2010.

[7] Nokia Energy Profiler. http://www.developer.nokia.com/Resources/Tools_and_downloads/Other/Nokia_Energy_Profiler/Quick_start.xhtml [Accedido septiembre 2012]

[8] PowerTutor.
https://play.google.com/store/apps/details?id=edu.umich.PowerTutor&hl=es
[Accedido septiembre 2012]

[9] Android Developers.
http://developer.android.com/index.html [Accedido año 2012]

[10] Canalys, Boletín de prensa 3 Febrero
2012.http://www.canalys.com/static/press_release/2012/canalys-press-release-030212-smart-phones-overtake-client-pcs-2011_0.pdf [Accedido julio 2012]

[11] Android Power Management.
http://www.kandroid.org/online-pdk/guide/power_management.html
[Accedido julio 2012]

[12] Motlhabi, M.B,*Advanced Android Power Management and Implementation of Wakelocks*, 2008.

[13] Android-Scripting. *Scripting Layer for Android brings scripting languages to Android.*
http://code.google.com/p/android-scripting [Accedido junio 2012]

[14] Jordan, L., Greyling, P., *Practical Android Projects,*United States: Springer-Science+Business Media LLC, 2011, capítulo 5.

[15] Ferrill, P, *Pro Android Python with SL4A*. United States: Springer-Science+Business Media LLC, 2011, capítulo 1.

[16] Tanabian, M., *Mobile Device / Application Power Profiling: Testing and Design considerations in mobile Devices & Apps for power performance and battery life*, NextGen TechDNA, 2011.

[17] International Telecommunication Union,*Key Global Telecom Indicators for the World Telecommunication Service Sector.*http://www.itu.int/ITUD/ict/statistics/at_glance/KeyTelecom.html
[Accedido junio 2012]

[18] International Telecommunication Union, *ICT Facts and Figures*, Union, 2011.
http://www.itu.int/ITU-D/ict/facts/2011/material/ICTFactsFigures2011.pdf
[Accedido junio 2012]

[19] Wi-Fi Alliance.
http://www.wi-fi.org/ [Accedido junio 2012]

[20] Wi-Fi.
http://en.wikipedia.org/wiki/Wi-Fi [Accedido junio 2012]

[21] Chandra, P., Lide, D., *Wi-Fi Telephony: Challenges and Solutions for Voice over WLANs*, Amsterdan: Elsevier/Newmes, 2007, capítulo 4.

[22] Estándar IEEE 802.11
http://www.ieee802.org/11/ [Accedido junio 2012]

[23] 3G.
http://es.wikipedia.org/wiki/Telefon%C3%ADa_m%C3%B3vil_3G [Accedido junio 2012]

[24] UMTS.
http://www.umtsworld.com/technology/overview.htm [Accedido junio 2012]

[25] Radio Resource Control
(RCC).http://en.wikipedia.org/wiki/Radio_Resource_Control [Accedido julio 2012]

[26] Qian, F., Mao, Z.M., Wang, Z., Sen, S., Gerber, A., Spatscheck, O., *Characterizing radio resource allocation for 3G networks*, Proc. Internet Measurement Conference, 2010.

[27] Especificaciones HTC
Legend.http://dl4.htc.com/web_materials/Manual/HTC_Legend/HTC_Legend_Vodafone_ES_UM.pdf [Accedido julio 2012]

[28] Android Debug Bridge (ADB).
http://developer.android.com/tools/help/adb.html [Accedido agosto 2012]

[29] Iperf.
http://iperf.sourceforge.net/ [Accedido agosto 2012]

[30] Aplicación iPerf para Android
https://play.google.com/store/apps/details?id=com.magicandroidapps.iperf&hl=es [Accedido agosto 2012]

[31] Lin, W., Wise, J.A., *Precise Power Characterization of Modern Android Devices*, Carnegie Mellon University, 2010.

Anexos

Anexo A: Instalación y configuración del SDK de Android y el entorno de desarrollo

A continuación se describen los pasos necesarios para instalar y configurar el SDK de Android en un PC con sistema operativo Linux:

Paso 1. Requisitos del sistema para instalar el SDK

Antes de instalar el kit de desarrollo de herramientas de Android, es necesario asegurarse que se cumplen los requisitos mínimos para poder proceder con la instalación.

En primer lugar, es necesario disponer de uno de los siguientes sistemas operativos:

- Windows XP (32-bit), Vista (32- bit o 64-bit), o Windows 7 (32-bit o 64-bit).
- Mac OS X 10.5.8 o posteriores solo (solo x86).
- Linux (Ubuntu Linux, Lucid Lynx)
  - GNU C Library (glibc) 2.7 o posteriores.
  - En Ubuntu Linux, versión 8.04 o posteriores
  - Las distribuciones de 64-bit deben ser capaz de ejecutar aplicaciones de 32-bit.

Por otro lado, será necesario instalar un entorno de desarrollo, IDE (Integrated Development Environment), compatible con los sistemas operativos anteriores. Android recomienda utilizar el IDE de Eclipse, con los componentes que se detallan a continuación:

- Eclipse 3.6.2 (Helios)[7] o posterior. Hay distintos paquetes de eclipse disponibles para cada tipo de plataforma. Para el desarrollo de Android se recomienda instalar uno de estos paquetes.
  - Eclipse IDE for Java Developers
  - Eclipse Classic
  - Eclipse IDE for Java EE Developers

- JDK 6 (kit de desarrollo de Java).

- Android Development Tools plug-in (para poder desarrollar aplicaciones Android con Eclipse). Su instalación se explica más adelante.

Además, será necesario tener un espacio en disco suficiente para todos los componentes que se necesiten instalar.

Paso 2.Descargar el SDK Starter Package

Para descargar el SDK, nos dirigimos a la página para desarrolladores de Android[8], y seleccionamos el SDK adecuado para nuestro sistema operativo. Este paquete no es el entorno de desarrollo completo, sino que incluye una herramienta completa que permite descargar todos los paquetes del SDK, así como la última versión de Android.

Paso 3.Añadir plataformas y otros paquetes

Una vez descargado el SDK, vamos a instalar los paquetes necesarios para el entorno de desarrollo mediante la herramienta Android SDK Manager (ver Figura 32). A continuación se explica cómo se realiza la instalación en un sistema operativo Linux.

Abrimos un terminal y nos dirigimos al directorio *tools* del paquete descargado y en un terminal ejecutamos el comando `android`. Aparecerá una pantalla similar a la Figura 32 con todas las APIs de las distintas versiones de Android, junto con las herramientas de desarrollo necesarias. Seleccionamos los paquetes que deseamos descargar y los instalamos. Como mínimo será necesario descargar una de las versiones disponibles de Android y las herramientas bajo la carpeta *Tools*.

[7] http://www.eclipse.org/downloads/
[8] http://developer.android.com/sdk/installing/index.html

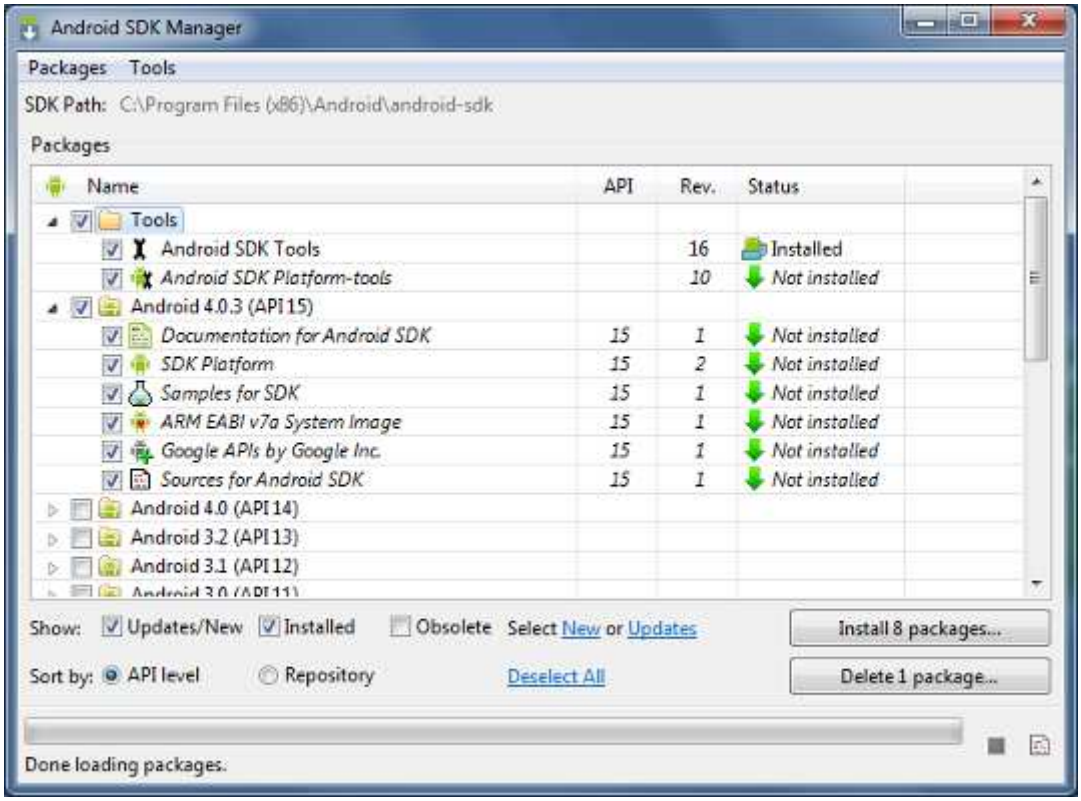

Figura 32. Herramienta Android SDK Manager

Una vez que se han descargado e instalado los paquetes necesarios, si no disponemos de un terminal físico en el que probar las aplicaciones desarrolladas, podemos crear un dispositivo virtual, también denominado emulador. Para ello abrimos el menú *Tools* en el Android SDK Manager y seleccionamos Manage AVDs. Aparecerá una nueva pantalla en la que se pueden crear, editar y eliminar dispositivos virtuales. Podemos crear varios emuladores con distintas versiones de Android, apariencia, tarjeta de memoria SD, y características hardware.

Paso 4. Instalar el plug-in ADT para Eclipse

Android ofrece un plug-in específico para el IDE de Eclipse, denominado ADT (Android Development Tools), que permite utilizar las herramientas del SDK para construir aplicaciones Android. ADT extiende las capacidades de Eclipse para poder crear de forma rápida y sencilla proyectos Android, aplicaciones con interfaz de usuario, depurar aplicaciones usando las herramientas del SDK, e incluso distribuir las aplicaciones creadas.

Este plug-in no está incluido en el SDK, sino que su instalación se realiza desde Eclipse. Para ello abrimos Eclipse y en el menú *Help* seleccionamos I*nstall new software.* Hacemos clic en *Add* y añadimos el repositorio de Android: https://dl-ssl.google.com/android/eclipse/.A continuación, dentro del repositorio seleccionamos el plug-in a instalar y completamos la instalación.

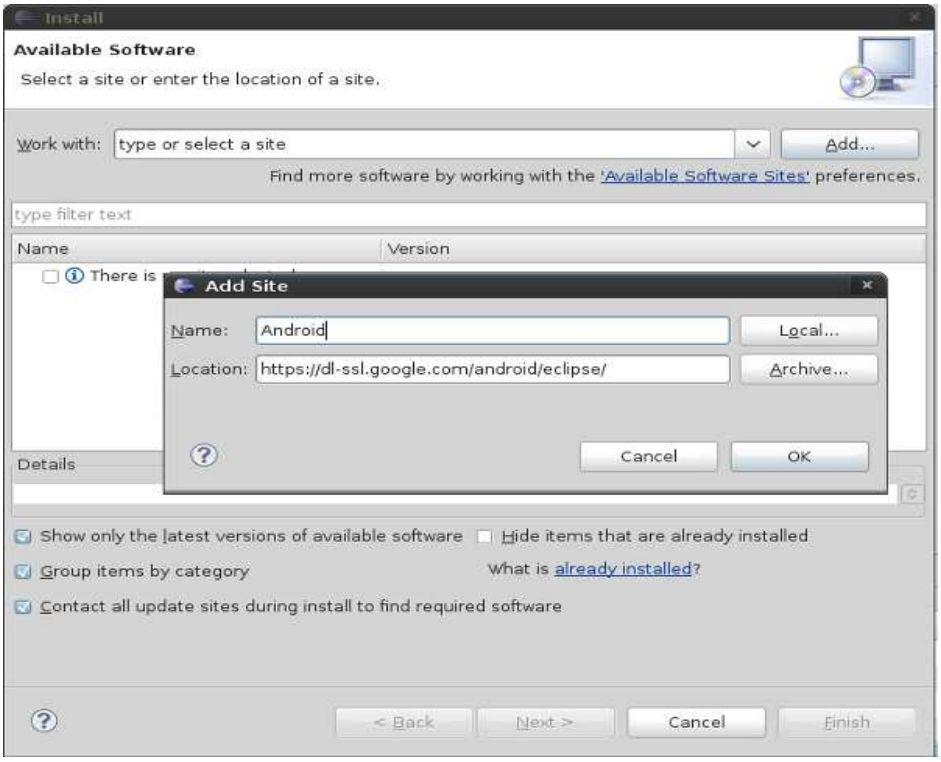

Figura 33. Configuración del plug-in ADT en Eclipse

Utilizar Eclipse con el plug-in ADT es la opción más sencilla y recomendada para el desarrollo Android, pero si se quiere trabajar con otro IDE, no es necesario instalar Eclipse o ADT. En su lugar, se pueden utilizar directamente las herramientas del SDK de Android para construir y depurar aplicaciones, aunque es mucho más complejo.

Anexo B: Herramienta ADB (Android Debug Bridge)

Para hacer uso de la herramienta de Android ADB, en primer lugar hay que configurar el acceso al teléfono desde el PC. Aunque Linux detecta el dispositivo desde el arranque, no lo configura de la forma necesaria para poder utilizar las herramientas del SDK. Es necesario crear una regla udev para que cada vez que conectemos el dispositivo, el sistema operativo lo reconozca como un terminal Android, y podamos acceder directamente a él con la herramienta ADB. Para ello editamos como root el archivo*/etc/udev/rules.d/99-android.rules*, creando la siguiente regla:

```
SUBSYSTEM==”usb”, ATTRS{idVendor}==”0bb4”,
SYMLINK+=”android_adb”, MODE=”0666”
```

El único dato que depende del terminal que se conecte, es el `idVendor` que es el identificador del fabricante. Para saber cuál es el de nuestro dispositivo, lo conectamos y ejecutamos el comando `lsusb`. Con este comando nos aparece información de todos los puertos USB y los dispositivos conectados. Buscamos el nuestro, que tendrá un aspecto similar al indicado en la Figura 34, cambiando el fabricante del dispositivo. El idVendor que necesitamos son los 4 primeros números del ID.

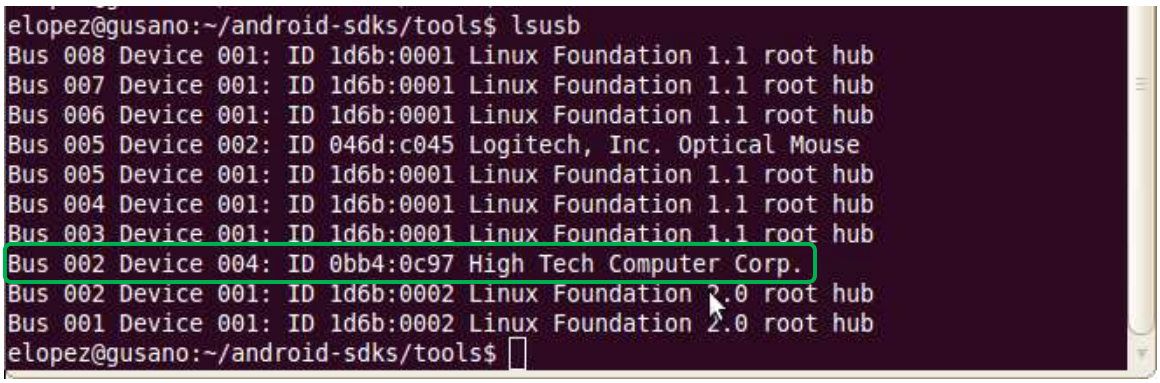

Figura 34. Comando lsusb para el idVendor de un dispositivo Android

Una vez creada la regla udev ya podemos acceder al dispositivo, tanto desde línea de comandos como desde el ADT de Eclipse.

Antes de conectar el dispositivo para trabajar con él, tenemos que activar la opción “Depuración de USB”, que se encuentra en el menú *Ajustes/Aplicaciones/Desarrollo* del teléfono. Es importante que cuando conectemos el teléfono nunca se active la opción “Almacenamiento USB” o “Unidad de disco”, ya que no es compatible con el modo de depuración.

Podemos comprobar si nuestro dispositivo ha sido reconocido, ejecutando el comando `adb devices`, que nos mostrará el número de serie de los terminales Android conectados al sistema (ver Figura 35).

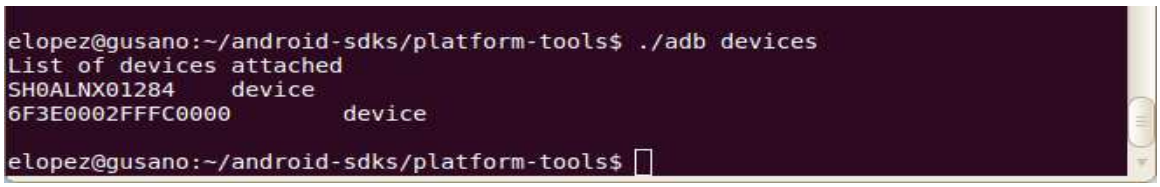

Figura 35. Respuesta al comando adb devices cuando hay dos dispositivos conectados

Si hay varios dispositivos conectados, para realizar operaciones con cada uno de ellos, indicaremos cual es el nuestro de la siguiente forma:

```
adb  -s [S/N] [command]
```

Donde `S/N` es el número de serie que hemos obtenido con el comando `adb devices` y `command` es la operación que queremos realizar sobre el terminal.

Con la herramienta ADB podemos realizar un gran número de operaciones como abrir una shell, copiar archivos desde/hacia el terminal, instalar/desinstalar aplicaciones, reiniciar el dispositivo, etc... Las más importantes son las siguientes:

Tabla 10. Operaciones principales con la herramienta ADB

Operación	Descripción
devices	Muestra el número de serie de todos los dispositivos Android conectados
shell	Abre una shell del dispositivo que se indique
push archivo_local archivo_destino	Copia un archivo desde el ordenador al dispositivo
pull archivolocal archivo_destino	Copia un archivo desde el dispositivo al ordenador
install aplicacion.apk	Instala una aplicación en el dispositivo
uninstall aplicacion.apk	Desinstala una aplicación de dispositivo
reboot	Realiza un *reboot* del dispositivo
remount	Remonta la partición /system del dispositivo para lectura/escritura
help	Muestra la ayuda de la herramienta ADB junto con la lista completa de todos los comandos

El comando `adb shell` es uno de los más importantes puesto que nos permite trabajar con el teléfono por línea de comandos tal y como si estuviéramos trabajando con un ordenador con sistema operativo linux. Una vez ejecutado `adb shell` podemos introducir comandos como `ls, ping, ifconfig, cat, chmod`...

Anexo C: Instalación del Scripting Layer para Android

La aplicación SL4A no se encuentra disponible en el mercado de aplicaciones de Android por lo que hay que descargarla de la página del proyecto de Google.[9] Como se va a instalar una aplicación que no procede del sitio oficial aplicaciones, hay que asegurarse que la opción *Fuentes desconocidas* en el menú del teléfono *Ajustes/Aplicaciones* se encuentra habilitada. Podemos instalar la aplicación descargándola desde el ordenador e instalándola mediante la herramienta ADB en el dispositivo, o bien reconociendo el código BIDI que aparece en la página del proyecto.

Una vez instalada la aplicación SL4A, la abrimos y en el menú *View* seleccionamos *Interpreters*. Por defecto se ha instalado Shell, pero podemos instalar otros los lenguajes de scripting. Como en este proyecto se ha trabajado con Python, ha sido el intérprete que se ha instalado. En el menú seleccionamos añadir un nuevo intérprete y se escoge el que se desee instalar, en nuestro caso Python 2.6.2.

Figura 36. Selección del lenguaje a instalar en la herramienta SL4A

Automáticamente, se inicia la descarga del paquete de Python y una vez descargado, lo abrimos para realizar la instalación. Cuando ya está instalado si abrimos el intérprete, podemos comenzar a escribir comandos.

[9] http://code.google.com/p/android-scripting/

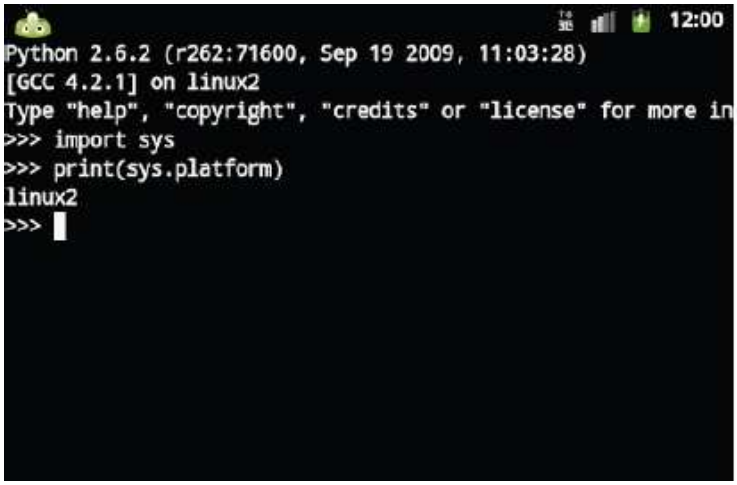

Figura 37. Intérprete de comandos para Python en la herramienta SL4A

En el menú *Scripts* de la aplicación, se puede apreciar que se han añadido algunos scripts Python de ejemplo. Los scripts se sitúan en la carpeta */sdcard/sl4a/scripts*de la memoria externa SD. Para poder ejecutar nuestros scripts los debemos guardar en esta carpeta. Podemos crear scripts en el propio teléfono, lo cual es un poco incómodo, o en el ordenador. Si desde el ordenador desarrollamos un script para alojarlo en la carpeta adecuada podemos hacer uso de la herramienta ADB mediante el siguiente comando que copia el script desde el ordenador al teléfono.

```
adb push myscript.py /sdcard/sl4a/scripts/myscript.py
```

Anexo D: Acceso root al HTC Legend

Antes de comenzar este proceso, es necesario haber instalado el SDK de Android y Eclipse (ver Anexo A). Además, todos los pasos descritos se deben realizar desde un PC con sistema operativo Windows.

El primer lugar, hay que crear lo que se denomina una *goldcard*, que es una tarjeta microSD modificada que permiteevadir las restricciones de bloqueo CID[10] en el terminal. Para convertir la tarjeta microSD en goldcard se siguen los siguientes pasos:

1. Es necesario descargarse una herramienta que podemos en encontrar en el siguiente link: http://www.filefactory.com/file/b1d16ag/n/GoldCardTool_exe

2. Con la memoria microSD dentro del teléfono, nos dirigimos al menú *Ajustes>Almacenamiento en tarjeta* y seleccionamos *Retirar tarjeta SD* y a continuación *Formatear tarjeta SD*.

3. A continuación conectamos por USB el teléfono al ordenador en modo Depuración de USB. Abrimos el programa descargadocomo administrador y seleccionamos*get CID* (probar con MMC0 si MCC1 falla).

4. Seleccionamos el link que aparece para abrir el generador de goldcard y copiamos el CID de la herramienta. A continuación se recibirá un email

5. Ahora seleccionamos como tipo de conexión USB, Unidad de disco.

6. En el programa seleccionamos *Refresh* y seleccionamos nuestra tarjeta microSD.

7. Seleccionamos *Load goldcard.img* y cargamos el archivo que hemos recibido por email. Después seleccionamos *Patch MMC*.

Antes de continuar es necesario instalar el programa HTC Sync versión 2.0.33. Por otro lado, hay que instalar las herramientas necesarias para hacer el HTC Legend,[11] y descomprimirlas en el disco duro del PC. Continuamos siguiendo los siguientes pasos:

1. Apagamos el teléfono y lo encendemos manteniendo pulsado el botón de encendido y el de bajar volumen hasta que aparece la pantalla de Fastboot. Volvemos a mantener pulsado el botón de encendido hasta que la pantalla de Fastboot aparece con color rojo.

[10] El código CID identifica al proveedor del teléfono y define el nivel de protección cuando se accede desde el PC.
[11] http://www.4shared.com/zip/3mH_LFAM/r4-legend-root.html)

2. Conectamos por USB el teléfono al PC y abrimos el archivo *step1-windows.bat* de las herramientas que hemos descargado.

3. Una vez que vemos el menú aparece de nuevo en la pantalla del teléfono, navegamos a BOOTLOADER y pulsamos el botón de encendido, luego navegamos a RECOVERY, utilizando los botones de volumen para navegar y el botón de encendido para seleccionar. El teléfono debería reiniciarse.

4. Abrimos el archivo *step2-windows.bat* y esperamos a que termine. (Antes de realizar este paso hay que asegurarse de tener instalado HTC Sync versión 2.0.33).

5. Ahora deberíamos ver una pantalla más elegante de *recovery*. Con el botón óptico de navegación nos movemos a *Wipe* y lo seleccionamos pulsando dicho botón. Luego navegamos a *Wipe data/factory reset* y lo seccionamos.

6. Por último, pulsamos el botón de bajar el volumen para regresar al menú principal y seleccionamos *Flash zip* desde la tarjeta SD y seleccionamos *rootupdate.zip*. Una vez completado este paso, ya tenemos acceso root al HTC Legend.

Anexo E: Configuración del escenario de trabajo Wi-Fi

En este anexo sedescriben los pasos que se han seguido para configurar el escenario de trabajo con la tecnología de comunicación Wi-Fi. Además, se explica cómo se ha realizado la conexión a una red Wi-Fi mediante línea de comandos en un dispositivo Android. En la Figura 38, se vuelve a mostrar la red planteada para evaluar el consumo asociado a Wi-Fi.

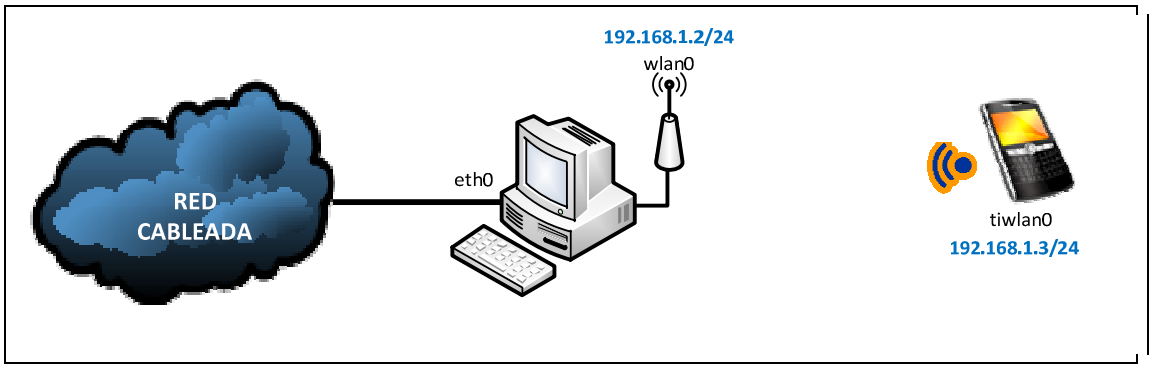

Figura 38. Escenario de trabajo con la tecnología Wi-Fi

Para conseguir que la interfaz inalámbrica del ordenador funcione como un AP, se comparte la conexión de Internet por cable del ordenador (eth0) con la interfaz inalámbrica wlan0. Para ello se han seguido los pasos que se detallan a continuación.

En primer lugar, debemos activar el renvío de paquetes, ya que normalmente un ordenador conectado a una red TCP/IP descarta todos los paquetes que le llegan que no son dirigidos a él. Podemos hacerlo explícitamente mediante el comando:

```
echo 1  > /proc/sys/net/ipv4/ip_forward
```

Esto también lo podemos conseguir modificando el archivo "/etc/sysctl.conf", añadiendo la línea `net.ipv4.ip_forward=1` para que se active el renvío de paquetes cada vez que arranquemos el ordenador

Por otro lado se debe activar el NAT, ya que se va a utilizar un direccionamiento privado. El comando necesario es el siguiente:

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

A continuación asignamos una IP privada y fija a la interfaz wlan0 y la habilitamos:

```
ifconfig wlan0 192.168.1.2 netmask 255.255.255.0 up
```

Para poder configurar la wlan0 como un AP necesitamos el paquete hostapd (si no está instalado lo podemos descargar con el comando `sudo apt-getinstallhostapd`). Hostapd es un *daemon* de espacio de usuario que permite configurar puntos de acceso. En el fichero "/etc/hostapd.conf" se editan los parámetros del AP, de forma que se indica la interfaz, el driver, la ssid, el canal… Además, se puede configurar la autenticación de la red Wi-Fi, aunque en este caso se ha dejado la red abierta. Los parámetros más importantes que se han editado son los siguientes:

```
interface=wlan0
driver=nl80211
ssid=ELENA
channel=11
```

Para que surjan efecto los cambios realizados, reiniciamos el daemon hostapd:

```
sudo /etc/init.d/hostapdrestart
```

Por último, se configura el modo de la interfaz wlan0 como master para que funcione como AP:

```
iwconfig wlan0 mode master
```

En el terminal también tenemos que configurar la conexiónWi-Fi, ya que no podemos hacer uso del "Administrador de Conexiones", puesto que una vez establecida la conexión no nos permite modificar los parámetros del driver y es necesario establecer la conexión con el AP mediante línea de comandos.

En la siguiente tabla se indica el nombre de los drivers y módulos necesarios para habilitar la interfaz inalámbrica, junto con su localización en el sistema operativo Android.

Tabla 11. Módulos necesarios para habilitar la interfaz inalámbrica por línea de comandos

Modulo	Nombre del archivo	Localización
WLAN Driver	tiwlan_drv.ko	/system/lib/modules/
SDIO Driver	sdio.ko	/system/lib/modules/
WLAN CLI	tiwlan_cu	Accesible desde cualquier localización
WLAN Loader	tiwlan_loader	/system/bin/
INI File	tiwlan.ini	/system/etc/wifi

En primer lugar puesto que la conexión se va configurar mediante línea de comandos, abrimos una shell del terminal en modo root. A continuación cargamos los módulos de los drivers SDIO y WLAN con el comando insmod:

```
insmod /system/lib/modules/sdio.ko
insmod /system/lib/modules/tiwlan_drv.ko
```

Por otro lado se carga el resto del firmware necesario con el siguiente comando:

```
/system/bin/tiwlan_loader -f /system/etc/wifi/Fw1273_CHIP.bin -
e /proc/calibration -i /system/etc/wifi/tiwlan.ini
```

El siguiente paso es habilitar la interfaz inalámbrica y asignar una dirección IP fija privada:

```
ip link set tiwlan0 up
ipaddr add 192.168.1.3/24 dev tiwlan0
```

Indicamos que el *gateway* por defecto es el AP:

```
ip route add default via 192.168.1.2 dev tiwlan0
```

También tenemos que indicar cuáles son los servidores DNS. En Android no existe un archivo resolv.conf donde se encuentran los servidores DNS. Para indicar un servidor DNS lo realizamos con el siguiente comando.

```
setprop net.dns1 8.8.8.8
setprop net.dns2 6.6.6.6
```

Una vez introducidos todos los comandos anteriores podemos establecer la conexión con la red anunciada por el AP. Para ello utilizamos un programa de línea de comandos disponible en Android que permite realizar operaciones sobre el driver de la interfaz IEEE 802.11. Abrimos una shell del teléfono y ejecutamos el comando tiwlan_cu, de forma que nos aparece la interfaz de línea de comandos con un menú como se puede ver en la Figura 39. Para entrar en cada uno de los submenús se introduce la letra mayúscula que aparece en el nombre de cada uno de ellos. La secuencia para conectarnos a la red ELENA, sería la siguiente.

```
tiwlan_cu
/ as                # Inicio escaneo de redes
t                   #Stop escaneo
/ c c ELENA         #Conexión a la ESSID ELENA
```

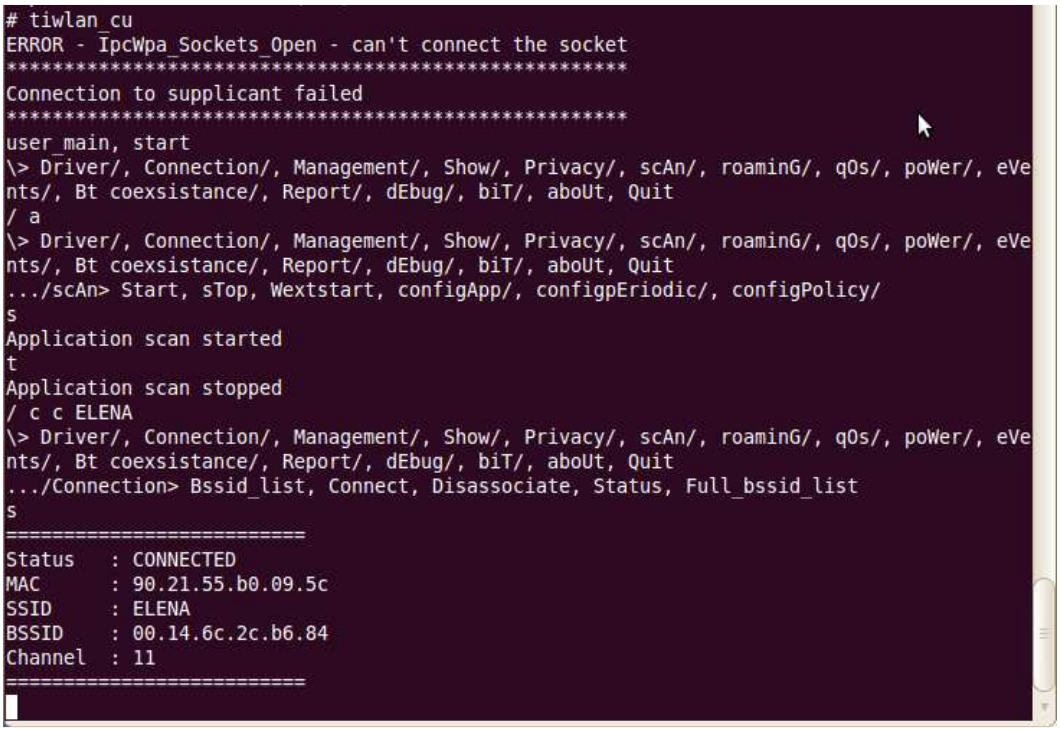

Figura 39. Interfaz de línea de comandos tiwlan_cu para establecer conexión con una red Wi-Fi

Anexo F: Código aplicación BatteryMonitor

A continuación se muestra el código fuente de la aplicación desarrollada para la monitorización del estado de la batería.

- **BatteryMonitor.java**

```
package pfc.uc3m.batterymonitor;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;

public class BatteryMonitor extends Activity {
   /** Called when the activity is first created. */

    @Override
    public void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

    Intent intent = new Intent("BatteryService");
    startService(intent);

            }
}
```

- **BatteryService.java**

```
package pfc.uc3m.batterymonitor;

import android.app.Service;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.BatteryManager;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.util.Log;
import android.os.IBinder;
import android.widget.Toast;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.io.File;
import java.io.FileOutputStream;
import java.io.OutputStreamWriter;
```

```
public class BatteryService extends Service {
    int level;
    int scale;
    float voltage;
    String date;
    String text;
    File f;
    OutputStreamWriter out;

    @Override
    public void onCreate() {
            super.onCreate();
            f = new File("/sdcard/bat_monitor_3Gidle.txt");
            registerReceiver( batteryReceiver, new
            IntentFilter(Intent.ACTION_BATTERY_CHANGED) );
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        Toast.makeText(this, "service starting", Toast.LENGTH_SHORT).show();
        return START_STICKY;
    }

    @Override
    public void onDestroy(){
            super.onDestroy();
            Toast.makeText(this, "service done",
Toast.LENGTH_SHORT).show();
    }

    @Override
    public IBinder onBind(Intent intent){
            return null;
    }

     public void updateBatteryCondition(int level, float volt){
            try{
                    int capacity = 1300*level/100;
                    date = new SimpleDateFormat("dd-MM-yyyy
                    HH:mm:ss").format(new Date());
                    text= date+" "+level+" "+capacity+" "+volt+"\n";
                    out = new OutputStreamWriter(new FileOutputStream(f,true));
                    out.write(text);
                    out.flush();
            }
            catch (Exception ex){
               Log.e("Ficheros", "Error al escribir fichero a tarjeta SD");
```

```
            }

        }

    private BroadcastReceiver batteryReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive( Context context, Intent intent ){

            scale = intent.getIntExtra(BatteryManager.EXTRA_SCALE,-1);
            level = intent.getIntExtra(BatteryManager.EXTRA_LEVEL, -1
            )*100/scale;
            voltage =
            (float)intent.getIntExtra(BatteryManager.EXTRA_VOLTAGE, -1) /
            1000;
            updateBatteryCondition(level,voltage);

    }
    };
}
```